Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática,
2011

Flávio Leonel Paradinha Lopes

Processamento com Memristors

Flávio Leonel Paradinha Lopes

Processing With Memristors

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e de Telecomunicações, realizada sob a orientação científica de Prof. Dr. José Manuel Neto Vieira, Departamento de Electrónica, Telecomunicações e Informática, Universidade de Aveiro; de Prof. Dr. Rui Manuel Escadas Ramos Martins, Departamento de Electrónica, Telecomunicações e Informática, Universidade de Aveiro.

o júri / the jury

presidente / president	**Prof. Dr. Alexandre Manuel Moutela Nunes da Mota** Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro
orientador / adviser	**Prof. Dr. José Manuel Neto Vieira** Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro
co-orientador / co-adviser	**Prof. Dr. Rui Manuel Escadas Ramos Martins** Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro
arguente / examiner	**Prof. Dr. João Paulo de Castro Canas Ferreira** Professor Auxiliar do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

agradecimentos / acknowledgements

Prestes a concluir mais uma importante etapa da minha carreira académica, que chegou a esta fase não apenas pelo meu trabalho e dedicação, mas também pelo apoio que me têm manifestado todos aqueles que me têm acompanhado neste últimos anos, é com enorme satisfação e orgulho que aqui deixo expressos os meus sinceros agradecimentos a todos os que, directa ou indirectamente, contribuíram para a consecução deste projecto e me incentivaram ao longo deste tão sinuoso percurso.

Em primeiro lugar, e com um carinho muito peculiar, manifesto a minha eterna gratidão aos meus pais, pelos esforços que fizeram ao longo da vida para que fosse possível chegar a esta fase, pelo apoio incondicional que sempre me dedicaram, sem o qual, certamente, eu não estaria aqui a finalizar esta dissertação. A eles dedico este meu trabalho e agradeço o terem-me facultado as ferramentas indispensáveis para eu poder construir o meu futuro.

O meu especial agradecimento ao meu orientador, Professor Doutor José Neto Vieira e ao meu co-orientador Professor Doutor Rui Escadas Martins, pelo tempo e disponibilidade que sempre me dispensaram, pela sua compreensão e profissionalismo e, sobretudo, por terem aceite este meu desafio e, também, me terem feito acreditar que eu tinha as capacidades e conhecimentos necessários para desenvolver e levar este projecto até ao fim.

A todos os meus amigos e colegas que me ajudaram na integração nesta nova cidade e tornaram a vida longe de casa mais agradável.

E por fim, mas com uma particular importância, à Elisabete pela paciência que teve comigo nos momentos em que me afastava e me isolava para trabalhar e quando as coisas corriam menos bem era ela que suportava o meu mau humor

palavras-chave Memristor, modelo SPICE do memristor, combinação linear, computação analógica, memória analógica

Resumo

Em engenharia procura-se sempre melhorar o produto anterior. Torná-lo mais eficaz, mais rápido, enaltecer a sua funcionalidade. Entretanto chega-se a um ponto onde o desenvolvimento se torna muito complexo e dispendioso. Chegados a esse ponto é necessário procurar novos métodos para continuar o desenvolvimento desse produto.

Numa dessas procuras foi descoberta a realização física de um novo componente electrónico, o memristor. Este novo componente compromete-se a revolucionar o mundo electrónico, pois promete ser usável num leque bastante alargado de aplicações.

O memristor promete ser um elemento de memória não volátil, pois consegue fixar o seu valor resistivo mesmo não estando a ser alimentado. Dado que o estado interno de um memristor se traduz num valor resistivo, pensa-se que um memristor pode ser usado para realizar combinações lineares, em que os coeficientes variam ao longo do tempo, ao invés de serem fixos, como era a única maneira de fazer até agora.

Nesta dissertação pretende-se provar que esta aplicação de memristores é possível. Para tal ser provado, é criado o modelo de um memristor ideal, que é utilizado para simular os circuitos que provam se é, ou não, possível implementar a função pretendida. É também desenvolvido um circuito usado para forçar, ou ler, um estado do memristor, sem ser necessária qualquer preocupação com as características físicas deste componente.

Após se obter o modelo, são discutidos os resultados obtidos, por forma a comprovar que estes são satisfatórios e que se pode continuar o trabalho. O mesmo processo de validação é usado com o programador. Dado que o modelo e o programador funcionaram como esperado, é possível passar ao âmago desta dissertação, criando-se o circuito necessário para se provar se é, ou não, possível implementar a operação desejada.

Findo este trabalho, conclui-se que os memristores podem, de facto, ser usados para a aplicação pretendida, pelo que se abrem muitas portas para trabalhos futuros, por forma a implementar e testar esta solução em várias aplicações.

keywords Memristor, memristor SPICE model, Linear Combination, Analog Computation, Analog Memory

Abstract

In engineering we are always trying to make a product better. Making it faster, more efficient, adding new and better features. Meanwhile we reach a point where that product's development becomes too complex and expensive. When that point is reached it is necessary to search for new methods to continue the product's development.

In one of those searches the implementation of a new device was found, the memristor's. This new device promises to revolutionize the electronic's world, since it is expected to be useful in a wide range of applications.

The memristor has potential to be a non-volatile memory element, since it holds its resistive value, even after it is unplugged from a power source. Given that the memristor's inner state is translated into a resistive value, it is possible to think that memristors can be used to perform linear combinations, where the coefficients will change along the time, between iterations, instead of being fixed, like it was the only way, until now.

In this thesis we aim to prove that this memristor's application is possible. For that to be proven, an ideal memristor model is created, in order to simulate the circuits that will prove, or not, that the application is realizable. A circuit that forces a state, or reads one is also developed, in order to provide an abstraction of the device's physical characteristics.

Right after the model is obtained, we discuss the simulation's results, in order to decide if we can keep with the work or not. The same approach is used with the programmer. Being that the model and the programmed worked as expected, we will now carry on to the fulcra point of this thesis, hence creating the necessary circuitry in order to prove our point.

At the end of this task, we conclude that memristors can actually be used to perform the desired operation. With this being proved lots of works appear, to test and implement the proposed solution in other applications.

Contents

List of Figures

List of Tables

Acronyms

AC	Alternate Current
ADALINE	Adaptive Linear Neuron
ANN	Artificial Neural Network
CMOS	Complementary Metal-Oxide-Semiconductor
DC	Direct Current
FIR	Finite Impulsive Response
FPGA	Field Programable Gate Array
HP	Hewlett Packard
MRAM	Magnetoresistive Random Access Memory
PES	Polyethersulfone
RAM	Random Access Memory
RRAM	Resistive Random Access Memory
SPICE	Simulation Program with Integrated Circuit Emphasis

Chapter 1

Introduction

Back in May 2008, at Hewlett-Packard research labs, a team announced the discovery of the physical implementation of the fourth fundamental electrical device, the memristor. This device was first predicted by Chua, Leon [1] almost forty years ago, and belongs to the non-linear circuit elements.

This device has memory, it is not volatile, and can store a continuous value between the two boundaries, hence it is possible to think immediately in an endless number of applications. This thesis is based in one of those applications, and will be explained in section 1.1.

1.1 Motivation

From many years ago, engineers are trying to perform some operations in the digital world, faster and less power hungry. This lead them to search for analog implementation of those operations.

Some of the operations that were critical in the digital domain could never be implemented in the analog world, such as linear combinations, because they were too dynamical and there was no way to achieve that level of dynamism with conventional discrete analog components.

When back in May of 2008 the implementation of the memristor was presented, that piece of dynamism appeared undercover. With a component that has the kind of characteristics of a memristor, we can achieve the desired level of dynamism to implement whatever operation we need.

This was what lead us to this thesis. We are trying to prove that memristors can be used to implement that kind of operations. Operations such as linear combinations that can change its coefficients between iterations, that is our main goal in this thesis.

This is our main goal because that operation is the basis to implement whatever operation we need, that weren't yet achieved, and this could mean a huge leap forward in a tremendous variety of applications.

1.2 Objectives

This thesis has three main objectives. Provide a better understanding of this new device, create a system that is capable of performing linear combinations in hardware, and provide a basis for future work.

1.3 Contributions of this Thesis

The research performed in this work resulted in the following contributions:

- A better understanding of the memristor;
- An architecture for realization of linear combinations.
- A basis for future work with memristors

1.4 Outline

This thesis is structured as follows. In Chapter 2, we do an introduction to the device itself, we talk about the history of this device, we give a little insight into the types of memristors that exist or are under research, then we give a brief explanation on some of the most known research areas that are trying to adopt the memristor, and finally we explain thoroughly the physical principles behind the memristor. Following in Chapter 3 we will describe a memristor of the type that we considered on this text, and how to obtain a model to use in circuit simulators, and the we test the model. In Chapter 4 we create a programmer in order to create an abstraction of the different process that is needed, and discuss how it is possible to create a memory based on a matrix of memristors. Then in Chapter 5 we conclude our work by creating a circuit that is capable of performing linear combinations, while between operations, the coefficients change. Finally, in Chapter 6 we talk about what we achieved with this thesis and provide some guidelines for future research.

Chapter 2

Insights into the memristor

In this chapter we will introduce this device as extensively as possible. First we will introduce the device's history, from the early publications about a device like this, to the physical implementation. Then we will talk about the state-of-the-art.

Along this text we will use some variables that describe units or parameters. They are all explained on the table bellow:

Table 2.1: Variables and Parameters used

Symbol	**Interpretation**	**Unit**
φ	Magnetic Flux	Webber
V	Voltage across a device	Volt
t	Time	second
q	Charge	Coulomb
I	Electric current flowing through the device	Ampere
R	Resistance to the current flow of the device	Ohm
C	Capacitance of the device	Farad
L	Inductance of the device	Henry
M	Memristance of the device	Ohm
W	Memductance of the device	Siemens
w	Internal state of the device/ width of the dopped region	meter
D	Height of the Memristor	meter
R_{ON}	Memristor's resistance when $w = D$	Ohm
R_{OFF}	Memristor's resistance when $w = 0$	Ohm
v_d	Ion drift velocity	m/s
μ	Ion mobility ($\mu_{TiO_2} = 10^{-14}$)	m^2/(V.s)
E	Electric field intensity	V/m
H	Magnetic field intensity	A/m
B	Magnetic flux density	Wb/m^2
J	Electric current density	A/m^2
D	Electric flux density	C/m^2
ρ	Electric charge density	C/m^3

2.1 Memristor's History

After the presentation of a memristor's physical realization, we are now able to look back in time and analyze some curious published results, as well as a similar device.

Back in 1961, when Widrow et al. were introducing their adaptive neuron (ADALINE), they created a device, which they called *memistor* (in [8]). It was a three terminal device in which the resistance, across two of the terminals, was given by the current's time integral on the other terminal. It's working principle is much like the one in a transistor, with the difference of the current's time integral versus the instantaneous value.

This was an electrochemical device that changes, and stores, it's value based on electroplating. It consists of a conductive substrate with insulated connecting leads, and a metallic anode, all in an electrolytic plating bath.

They claim that they have built an analog capable memory element, capable of storing thousand of possible values, in which the values range from 100Ω to 1Ω, and take about $10s$ to switch from one end to the other.

The most curious thing was the *programming*, and sensing method. The programming was accomplished by applying a DC signal to the device, while the reading was made applying an AC signal, exactly like Chua's memristor [1][8].

Looking back to the *memistor* history, we note that it wasn't successful, maybe for it's electrochemical nature, which could not be used in integrated circuits, the growing technology of that time, therefore fading in relevance.

Some years after the *memistor* introduction, a paper appeared, reporting a switching phenomena in titanium dioxide thin films. This paper [9] was presented by Argall, from the Chelsea College of Science and Technology, in 1967. Once more, looking back with today's knowledge, we now understand that they were seeing memristance, even before it's existence has been introduced. It is relevant to notice that this paper take us to previous works, where they try to explain this switching effect in other materials. This is why this paper is interesting. This paper was the first that tried to understand this phenomena, that was exactly the same, but could not be explained by the same reasons as the ones before. Given that it is fair to say that this time was the very first time where someone talked/thought about the concept of memristance, even though not knowing what it was at that time.

It is quite curious that Chua and Argall never looked into each other's work, but as they are from "non-related" areas, it is understandable.

The memristor's story keeps getting written when in 1971 Chua publishes it's paper entitled "Memristor – The missing circuit element", where he wrote what we describe in section 2.4. Then again, in 1976, when he generalizes its concept to a broader range of devices and systems, the so called "memristive systems", which were used to provide more accurate models of some systems, and we also discuss that on section 2.4.

After all these happenings in such short time, the memristor's history remained idle for a long period of time. It was only in May of 2008 that another paragraph of this device's history was written. A HP's researching team published a paper in nature ([3]), that was about to change the electronics world. Finally a physical implementation of a memristor was discovered.

2.2 Types of Memristors

After looking back, and analyzing some device property's, we are now able to identify several cases of memristance that have been observed throughout the time.

There are two types of memristors, the molecular and ionic thin film devices, and the spin based and magnetic memristive systems, which will be explained below.

2.2.1 Molecular and Ionic Thin Film Memristors

These types of memristors are based on the different thin film crystal's properties, which exhibit hysterethic behavior when charge is applied to them. This type of devices can be divided into multiple sub-types, that we will describe right away.

Titanium Dioxide Memristors

These memristors are the ones where the memristance effect was observed, and published, by HP Labs, consequently it is the type that has more research and publications. We will focus on this type of devices too.

The titanium dioxide memristor is accomplished by layering stoichiometric titanium dioxide on top of positively doped titanium dioxide, in between two platinum electrodes. The positively doped region, since it has free charge carriers, acts as a conductor, while the stoichiometric layer acts as an insulator. The line that divide's the two layers, moves according to the device state, and this is the fact that is responsible for the memristance effect.

The overall resistance is given by the sum of the two layer's resistance.

Polymeric (ionic) Memristors

Polymeric memristors explore dynamic doping of polymer and inorganic dielectric type material to attempt and provoke hysterethic behavior. One component of the structure is free to move as a charge carrier, it could be the cationic or the anionic.

A single passive layer, between the active thin film and the electrode, attempt to exaggerate the extraction of ions from the electrode.

Graphene-Oxide Memristor

Two terminal devices consisting of graphitic sheets grown on nano cables or transferred onto a silicon oxide substrate exhibited an enormous and reversible nonvolatile memory effect, which was attributed to the formation and breaking of carbon atomic chains. Recently, reliable and reproducible resistive switching of graphene-oxide thin films and conjugated polymer functionalized graphene-oxide films were reported [7].

A trilayer device, in which we stack a layer of aluminum, one of graphene-oxide film and finally another of aluminum, with a cross-point structure was made on flexible polyethersulfone (PES) substrate with a uniform graphene-oxide film prepared by spin-casting. This device showed reliable and reproducible bipolar resistive switching with an on/off ratio of ~ 100, a retention time of longer than $10^5 s$, and switching voltages of $\sim 2.5V$.

Manganite Memristive Systems

Back in 2001, a substrate of bilayer oxide films based on manganite, describing memristive properties, has been shown, at the University of Houston.

Resonant-Tunneling Diode Memristor

Some certain types of quantum-well diodes, with special doping designs, have been shown to exhibit memristive properties.

2.2.2 Spin Based and Magnetic Memristive Systems

This type of memristors rely on the degree of freedom in electron spin. The electron spin polarization is altered through the magnetic "domain" wall, separating polarities. This allows the occurrence of hysterethic behavior.

Spin Torque Memristors

Some labs, including Seagate [7], are developing spintronic memristors.

These devices, much like the titanium dioxide memristors, change its state by altering the ratio between two types of material. This state alteration is achieved by the use of magnetization, to change the spin directions in one of the two regions of the device. These two regions are kept separated, by using s moving "wall", that is controlled by magnetization.

The overall resistance is given by the ratio of the two regions.

Spin Torque Transfer Magnetoresistive Random Access Memory

The development of MRAM has shown, in some cases, memristive properties.

The spin valve, the simplest configuration for a MRAM bit, allows the state to change. The spin valve consists of two (or more) conducting magnetic materials, that when they have the same spin moment, it is in the high conductance state, when the spin moment of the different materials, is opposed, the device is in the low conductance state. This type of devices, since the materials that make them, always have different magnetic properties (one soft, one hard, or so), which gives this device the hysterethic behavior.

The resistance is, consequently, controlled by a spin torque, which is induced by a current flowing through a magnetic junction, and is dependent on the difference in spin orientation between the two sides of the junction.

Depending on the material used in these devices, they can exhibit both ionic and magnetic properties. Sometimes they are referred to as "second-order memristive systems".

2.3 Applications

Nowadays there isn't yet any market product that uses memristors, but there are some on the horizon. We will now explain how memristors would be important in any of the applications that may benefit more from this device.

2.3.1 Memories

Memristors can be used to store information indirectly. Since memristors are not capable of storing energy, they cannot store the information directly.

As the memristors have the ability to switch between two values of resistance, they can be used to store information, using these two states of resistance. This is an advantage, since it is possible to build a memory cell with just one device, achieving an huge storage density.

We referred to digital storage, but the memristors capacities as a memory element go beyond the digital world. It can change its value to any value between R_{ON} and R_{OFF}, so it can be used, in the same way as with the digital memory, to store analog values.[1]

Going back to the memories, wouldn't it be great to have a computer that has the operative system always loaded in the RAM, eliminating therefore the boot time wait? A computer that doesn't lose a byte in a power failure? Memristors can provide non-volatile RAM, thus making things like the ones mentioned before possible.

Beside RAM, memristors can be used for permanent data storage, much like as today flash memory cells, but with the advantage that it doesn't have to be a digital memory, it could be analog.

2.3.2 Field Programmable Gate Arrays

Field programable gate arrays (FPGAs) are used to build, with not much effort, application specific processors from scratch. They provide interconnected digital gates, where the connections between them are chosen by the programmers, implementing therefore what they need to build a complete system.

Developers only chose FPGAs to build a system, when they have to meet tight timing schedules, such as in telecommunications, signal processing, and so on, since they can have full control over the data-path of a specific function, getting therefore the maximum performance of the system. In some applications even FPGAs can be too slow, so there is the need to improve the performance of these devices.

We can use memristors to implement analog adders, where we can control the resistance electronically according to previous values, making it therefore valuable for FPGAs, since the digital equivalent takes a lot more of resources and time. This could be a great advantage if we will process large vectors of information, so all the multiplying and accumulating in a FPGA become quite complex.

Lets follow a practical example. If we want to implement a 200^{th} order FIR filter, we will going to have 201 multiplications and sums. Trying to make this work in the minimum time possible, requires 201 multipliers, with at least 64 bits each if we are working with a regular 32 bit word, and another bunch of adders with lots of bits too. If we look at the equivalent implemented with memristors, it only requires 201 memristors, and the necessary programming and reading equipment, which is much less than the digital solution.

[1] This is the goal of this thesis.

2.3.3 Artificial Neural Networks

An artificial neural network is used to solve complex problems with simple operations, such as the multiplication and sum. They are based on the biological neural networks, like our brain, or other animal that has cognitive capabilities.

Biological neural networks are made from neurons, which are the fundamental element of a neural network, dendrites, that capture the signal from synapses and transmit it to the neuron, axons, that transmit the signal from the neuron to the synapses, and finally synapses that make the connection between the extremities of an axon to a dendrite. One single neuron can have multiple inputs, and multiple outputs as well.

Artificial neural networks are modeled from the biological ones, described above, so they will have equivalents to all its components. Now we will describe the function of each component and how they are implemented artificially.

Lets start by the dendrites. Dendrites capture a fraction of the signal provided by the previous axon, that fraction depends on the synapse, and it can be modeled by a linear combination of the input signal with a coefficient, much like a digital FIR filter. Now we approach the neuron. The neuron simply sums the input values, and if the signal is sufficient, it produces a single output. This single output can be transmitted to several neurons, by the axon, so it is quite simple to model, as we just have to grab the neuron's output value. We have already talked about synapse's function, but to clarify their role in the network we will discuss them again. Synapses modify the input value by a weight that is defined according to each function that the neuron is implementing.

The learning process that all neural networks have to pass, in order to do something, is called learning or training. What this process does is to modify the synapse's values.

All these functions are implemented in an artificial neuron, leaving it with only simple inputs and outputs. An artificial neural network can have several neurons, organized in layers to accomplish the desired application.

A memristor behaves exactly as a synapse, changing its value into a new one according to its past, what makes them perfect to implement neural networks. Since an electronic analog adder is very simple to make, the only element that was missing, to build an analog version of an artificial neural network, was the memristor.

2.3.4 Logic Implementation

Memristors can be used to perform logic operations as well. These devices implement material implication, a fundamental logic operation [10].

That operation's truth table is the following:
So the function that it implements is:

$$f(x, y) = \bar{x} + y \tag{2.1}$$

As we can see in the table, the operation is represented by the symbol '$\rightarrow$', so it is written "$x \rightarrow y$" and it is read "x implies y", or "if x, then y".

Table 2.2: Implication Truth Table

x	y	$x \rightarrow y$
0	0	1
0	1	1
1	0	0
1	1	1

It could be proved that with implication and the false operation (despite the input value, it's output is always 0) one can implement all the other logic operations, providing therefore a complete logic basis.

In order to achieve this logic operation with memristors, a circuit like the one of the next figure, must be implemented.

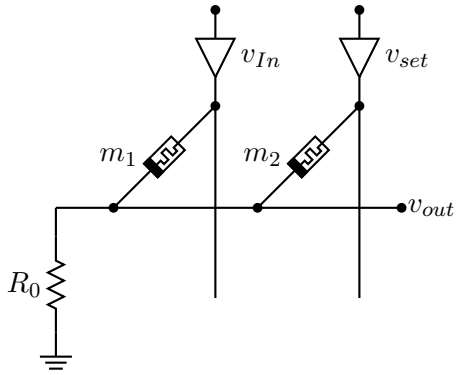

Figure 2.1: How to implement logic implication with memristors

Recalling that m_1 stores the value of x, m_2 stores y's value, and that implication's result will be stored in m_2 ($m_2 = m_1 \rightarrow m_2$), we will briefly explain how the implication is achieved.

In order to perform a logical operation, the values of x and y must be preloaded to m_1 and m_2 respectively. After the pre-charge of the values, a control signal is applied in both the inputs. According to the state of the memristors, and the value of R_0, the value of v_{out} will be pulled up or down, therefore changing the state of m_2. This state must later be sensed.

This construction can be scaled up to multiple inputs, outputs, and to perform a complete set of logic operations, as explained in [11].

2.4 Memristor

Back in 1971, Chua noticed that among the six possible relationships between two of the four basic electric variables (magnetic flux, charge, voltage and current), only five were known. The five shown below.

$$\delta\varphi = V\delta t \tag{2.2}$$
$$\delta q = I\delta t \tag{2.3}$$
$$\delta V = R\delta I \tag{2.4}$$
$$\delta q = C\delta V \tag{2.5}$$
$$\delta\varphi = L\delta I \tag{2.6}$$

From these relationships, Equation 2.4, Equation 2.5 and Equation 2.6 supply us with the basic well known circuit elements (resistor, capacitor and inductor, respectively), and the two remaining relationships provide us with the capability of calculating the voltage or current value from the magnetic flux or charge, respectively.

Hence the relationship that wasn't established yet, was the relationship between the magnetic flux and charge. It was then necessary to postulate this new device that provide us the missing relationship, shown below.

$$\delta\varphi = M\delta q \tag{2.7}$$

Since the magnetic flux is directly proportional to the charge, and recalling equations 2.2 and 2.3, we can conclude right away that the I-V relation will not be linear. With those equations still in mind, dividing both sides of the above equation by δt, and differentiating them, we obtain the following equation that gives us the voltage across a charge-controlled memristor.

$$V(t) = M(q(t))I(t) \tag{2.8}$$

Where

$$M(q) \equiv \frac{\delta\varphi(q)}{\delta q} \tag{2.9}$$

Similarly, the current in a flux-controlled memristor is

$$I(t) = W(\varphi(t))V(t) \tag{2.10}$$

Where

$$W(\varphi) \equiv \frac{\delta q(\varphi)}{\delta\varphi} \tag{2.11}$$

Notice that $M(q)$ has the unit of resistance (arrange equation 2.8 to confirm), so it will be called *incremental memristance*. The function $W(\varphi)$ has units of conductance, so similarly, it will be called *incremental memductance*.

A memristor at a given instant behaves like an ordinary resistor, but its value on that moment depends on the complete past history of the current. Notice that the value of the memristance (memductance) at an instant t_0 depends upon the time integral of the memristor current (voltage) from $t = -\infty$ to $t = t_0$. Given these properties, and in order to simplify the explanations and understanding of the device, henceforth we will describe it as if it were a resistance instead of memristance.

Going a bit further on the topic, we can change the device's resistance to the value we want, programming it therefore. If we want to read the information that was previously set on the memristor, we have to apply a signal (voltage or current) with null mean value, in order to assure that the time integral stays unchanged.

Given all these properties it is now possible to understand the choice of the name memristor for this device, as it has two functionalities, memory and resistance.

Chua postulated some theorems that we will not explain here, we will only mention one of them, that is quite important, the passivity theorem. It says that a memristor characterized by a differentiable charge-controlled $\varphi - q$ curve is passive if, and only if, its incremental memristance is nonnegative. If its incremental memristance is negative, this means that the device is providing power to the circuit, therefore it is an active device.

Five years after this concept presentation, Chua and his student, Kang, published a paper [6], broadening the concept of a memristor into a whole class of systems, the memristive systems. According to that paper a system can be called a memristive system if it is described by a system of equations like,

$$v(t) = R(w, i, t)i(t) \tag{2.12}$$
$$\dot{w}(t) = f(w, i, t) \tag{2.13}$$

where R is the system's resistance according to its internal state variable w, which dynamic is described by $f(w, i, t)$. Some examples of memristive systems were found, long before the memristor itself, such as an old tungsten light bulb. It is important to realize that the memristor, itself, is a particularization of these systems.

2.4.1 Electromagnetic Interpretation

As electrical engineers know, the fundamental devices can be understood by analyzing the *quasi-static* expansion of Maxwell's equations. The memristor is no exception, and these interpretations provide us with a deeper understanding of this new device. Chua [1] used this interpretation to suggest that a physical memristor realization was possible.

Let's begin by recalling Maxwell's equations in their differential form:

$$\nabla \times \mathbf{E} = -\frac{\delta \mathbf{B}}{\delta t} \tag{2.14}$$
$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\delta \mathbf{D}}{\delta t} \tag{2.15}$$
$$\nabla \cdot \mathbf{D} = \rho \tag{2.16}$$
$$\nabla \cdot \mathbf{J} = -\frac{\delta \rho}{\delta t} \tag{2.17}$$
$$\nabla \cdot \mathbf{B} = 0 \tag{2.18}$$

and that

$$\mathbf{D} = \varepsilon_0 \mathbf{E} + \mathbf{P} \tag{2.19}$$
$$\mathbf{B} = \mu_0(\mathbf{H} + \mathbf{M}) \tag{2.20}$$

Where $\mathbf{P}$ is the polarization, and $\mathbf{M}$ is the magnetization, ε_0 and μ_0 are the electric permittivity and magnetic permeability of free space, respectively.

It's possible to see that if we have a varying magnetic flux , we will have a varying electrical field, which generates a varying magnetic field, and *vice-versa*. If we want to calculate one of the fields, we will have to solve a system of equations, in order to have an accurate result. This is not needed if we are working with static fields, in which case we can obtain a solution by solving a simple equation, as the fields are independent.

In order to simplify the resolution of the system, we should look for a method that resembles the static fields resolution. To achieve that, we start by making a linear transformation, substituting t with

$$\tau = \alpha t \tag{2.21}$$

where α is a time-rate parameter. The time-rate parameter can be used to describe a family of functions τ, that have the same shape as the ones in t, but are compressed or stretched in time, depending on α being greater than 1, or otherwise, smaller. This parameter can be seen as a frequency variation parameter, in order to simplify the explanation. If we rewrite Eqs. 2.14 to 2.18, in order to include this linear transformation, we have:

$$\nabla \times \mathbf{E} = -\alpha \frac{\delta \mathbf{B}}{\delta \tau} \tag{2.22}$$

$$\nabla \times \mathbf{H} = \mathbf{J} + \alpha \frac{\delta \mathbf{D}}{\delta \tau} \tag{2.23}$$

$$\nabla \cdot \mathbf{D} = \rho \tag{2.24}$$

$$\nabla \cdot \mathbf{J} = -\alpha \frac{\delta \rho}{\delta \tau} \tag{2.25}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{2.26}$$

It is straight-forward that if $\alpha = 0$, the equations describe the static field solution, thus can be solved separately.

When we have a very small α, but still not equal to zero, the factors that will affect the system, will be attenuated by the small α, so $\mathbf{E}$ and $\mathbf{H}$ will not differ too much from those when $\alpha = 0$. It is, then, possible to say, that when α is small, the following is true

$$\mathbf{E} = (\mathbf{E})_0 + \alpha \left(\frac{\delta \mathbf{E}}{\delta \alpha} \right)_0 \tag{2.27}$$

$$\mathbf{H} = (\mathbf{H})_0 + \alpha \left(\frac{\delta \mathbf{H}}{\delta \alpha} \right)_0 \tag{2.28}$$

The subscripted zero indicates that the quantities in parentheses are evaluated for $\alpha = 0$. In this equation there is the static field solution, to which we added the variation of α.

The derivatives with respect to α in the equations above, can be obtained directly by the field laws, in fact, differentiating eqs. 2.22 through 2.26 in order to α, and setting $\alpha = 0$, it is possible to obtain the following expressions

$$\nabla \times \left(\frac{\delta \mathbf{E}}{\delta \alpha}\right)_0 = -\frac{\delta}{\delta \tau}(\mathbf{B})_0 \tag{2.29}$$

$$\nabla \times \left(\frac{\delta \mathbf{H}}{\delta \alpha}\right)_0 = \left(\frac{\delta \mathbf{J}}{\delta \alpha}\right)_0 + \frac{\delta}{\delta \tau}(\mathbf{D})_0 \tag{2.30}$$

$$\nabla \cdot \left(\frac{\delta \mathbf{J}}{\delta \alpha}\right)_0 = -\frac{\delta}{\delta \tau}(\rho)_0 \tag{2.31}$$

This leads us to what we wanted since the beginning, the chance to solve the equation without solving the system. It is easy to see that the terms that are differentiated in order to τ can be obtained by the static field equations.

If this approximation isn't satisfactory, it is possible to add square terms in α, proportional to the second derivatives, then cubic terms, and so on, until the approximation satisfies the requirements, always with $\alpha = 0$. If we do so, this will yield in a power series, like the following,

$$\mathbf{E} = \mathbf{E}_0 + \alpha \mathbf{E}_1 + \ldots + \alpha^k \mathbf{E}_k \tag{2.32}$$

$$\mathbf{H} = \mathbf{H}_0 + \alpha \mathbf{H}_1 + \ldots + \alpha^k \mathbf{H}_k \tag{2.33}$$

and recalling that

$$\mathbf{E}_0 = [\mathbf{E}]_{\alpha=0} \tag{2.34}$$

$$\mathbf{E}_1 = \left[\frac{\delta \mathbf{E}}{\delta \alpha}\right]_{\alpha=0} \tag{2.35}$$

$$\vdots$$

$$\mathbf{E}_k = \frac{1}{k!}\left[\frac{\delta^k \mathbf{E}}{\delta \alpha^k}\right]_{\alpha=0} \tag{2.36}$$

the procedure to obtain $\mathbf{H}_0$, $\mathbf{H}_1$, and so on, is similar to the one explained above for the various $\mathbf{E}_x$.

We can see that what was true for the first order derivatives, – the possibility to calculate the first order derivative, as being the static case solution – is still true for the higher order derivatives. It is possible to have a solution based on the previous degree iteration, recall from Equation 2.29 to Equation 2.31. Therefore it is possible to solve a field equation alone, without having to solve a system, and it can be done achieving the desired error.

A good result can be achieved by making a *quasi-static* approach to the problem, this is, considering only the zero and first order solutions. If we assume this and if $\alpha = 1$, Equation 2.32, and Equation 2.33, will yield

$$\mathbf{E} = \mathbf{E}_0 + \mathbf{E}_1 \tag{2.37}$$

$$\mathbf{H} = \mathbf{H}_0 + \mathbf{H}_1 \tag{2.38}$$

If we assume that all materials under consideration, are linear and isotropic, the constituent relations are simply given by

$$\mathbf{D} = \varepsilon\mathbf{E} \tag{2.39}$$
$$\mathbf{B} = \mu\mathbf{H} \tag{2.40}$$

And we should recall another important relation

$$\mathbf{J} = \sigma\mathbf{E} \tag{2.41}$$

With all this in mind, we can write the zero-order equations as:

$$\nabla \times \mathbf{E}_0 = 0 \tag{2.42}$$
$$\nabla \times \mathbf{H}_0 = \mathbf{J}_0 \tag{2.43}$$
$$\nabla \cdot \varepsilon\mathbf{E}_0 = \rho_0 \tag{2.44}$$
$$\nabla \cdot \mu\mathbf{H}_0 = 0 \tag{2.45}$$
$$\nabla \cdot \mathbf{J}_0 = 0 \tag{2.46}$$

and the first-order ones as

$$\nabla \times \mathbf{E}_1 = -\mu\frac{\delta\mathbf{H}_0}{\delta t} \tag{2.47}$$
$$\nabla \times \mathbf{H}_1 = \varepsilon\frac{\delta\mathbf{E}_0}{\delta t} + \mathbf{J}_1 \tag{2.48}$$
$$\nabla \cdot \varepsilon\mathbf{E}_1 = \rho_1 \tag{2.49}$$
$$\nabla \cdot \mu\mathbf{H}_1 = 0 \tag{2.50}$$
$$\nabla \cdot \mathbf{J}_1 = -\frac{\delta\rho_0}{\delta t} \tag{2.51}$$

If we analyze the above equations, we can see that there are three types of solutions. We can have solutions of the electric type, characterized by the absence of a zero-order magnetic field, and solutions of the magnetic type, characterized by the absence of a zero-order electric field, and these solutions only appear when there is no coupling between electric and magnetic fields, i.e, the conductivity is infinite. And finally, we have the solution where we have the zero-order electric field, and a zero-order magnetic-field, coupled by currents in a material that doesn't have infinite conductivity.

If we have the electrical field, and the magnetic field coupled, it should, somehow, be possible to relate them. Electric and magnetic fields, cause/are caused by voltage and current respectively, and we can think in at least three ways to relate voltage and current. The definitions of the three basic electrical components, the resistor, capacitor and inductor, and their voltage current relationship is the one mentioned in Equation 2.4, Equation 2.5, and Equation 2.6, respectively.

So let's start by the resistor. On a resistor, the first-order fields are negligible, so if we apply a zero-order electric field, that results on a current density, as shown in Equation 2.41

and that creates a zero-order magnetic field as stated by Equation 2.43, and this is the only way that this combination (first-order fields negligible) can be related.

Now if we think that we have a varying zero-order electric field, a negligible first-order electric field, and a zero-order magnetic field, we will have a non-negligible first-order magnetic field. If we have a varying zero-order electric field, we are considering Equation 2.48, where $\mathbf{J}_1$ comes from Equation 2.51, which takes us to Equation 2.44 to complete the relationship, and to have $\mathbf{H}_1$ in order to $\mathbf{E}_0$, only. When we think in what this relationship means, we conclude that a component that has a relationship where the variation of the voltage yields in a current, is a capacitor, as can be seen in Equation 2.5.

From the three basic components, the one that we didn't talk yet is the inductor. The inductor relates the zero-order magnetic-field with the first-order electric field, hence the negligible fields are the first-order magnetic field, and zero-order electric field. Here we grab Equation 2.47, substitute $\mathbf{H}_0$ from Equation 2.43, and we have a relationship that results in a first-order electric field that results from a zero-order magnetic field, and to make the connection to Equation 2.6, the magnetic field is created by the current density $\mathbf{J}_0$.

We talked about the fields relationships when both first-order, and only one first-order, and one zero-order fields are negligible. We now should talk about the case when both the zero-order fields are negligible, hence only the first-order fields are taken in account. First we should be aware that is only when we take in account a first-order field, that a field generates the other one, so when we take in account both first-order fields, we should expect a non-linear dynamic behavior, as mentioned earlier.

When we take in account only the first-order fields, we are looking at Equation 2.48 where the term that mentions $\mathbf{E}_0$ is neglected, and $\mathbf{J}_1$ comes from Equation 2.41. This relationship is very similar to the one presented by the resistor, but has a major difference. Here we are considering only the first-order fields, while when talking about the resistor we considered only the zero-order fields. This relationship is non-linear, as we said above. The only known component that seems like a resistor, but has a non-linear behavior is the memristor, and this is his fields relationship.

Chapter 3

Memristor Simulation Model

We have introduced the basics of the memristor, the physics behind it and how it works. We presented before the relationship between voltage and current, but that isn't enough to start a circuit simulator and start designing a circuit with memristors. Hence now we will describe how to create a simulation model, that will let us observe the memristor's behavior in a circuit.

3.1 Device

If we want to create a simulation model for a device, we must know the physical implementation of that device, in order to understand the physics behind its working principle. Here we will introduce the physical implementation of a *Titanium Dioxide* memristor, like the ones HP Labs introduced back in May of 2008.

The device that Stanley Williams presented is achieved by making a sandwich of *titanium dioxide* (TiO_2) between two *platinum* electrodes. The layer of TiO_2 is divided in two regions, one is stoichiometric TiO_2, while the other is constituted by positively doped titanium dioxide (TiO_{2-x}). The stoichiometric portion of the device is a pure semiconductor, working as an insulator, while the doped portion works as a conductor. This construction is shown in Figure 3.1.

Since we have two regions with different resistances, we can describe the total resistance by the sum of two resistances in series, while its value is given by the relationship between the region thickness over the total thickness.

Recalling that a memristor is a particularization of memristive systems, thus being described by the system of Equation 2.12 and Equation 2.13 it is possible to start understanding how the device works. The device's resistance according to its state is given by

$$R(w(t)) = R_{ON}\frac{w(t)}{D} + R_{OFF}\left(1 - \frac{w(t)}{D}\right) \tag{3.1}$$

where w is the frontier between the two regions of TiO_2 and D is the overall TiO_2 thickness, as shown in Figure 3.2.

As the current flows through the device, the doped region gets thicker or thinner, depending on the amount of current flowing and the device's past, changing its state and therefore

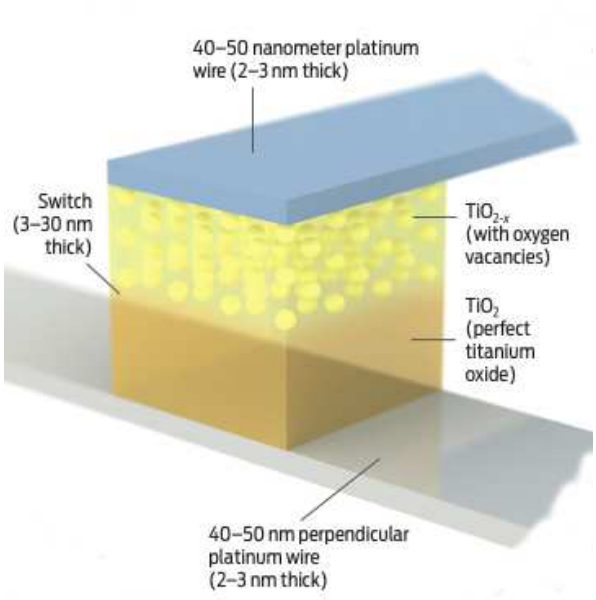

Figure 3.1: Memristor implementation from [2]

its equivalent resistance of the doped and undoped regions. As this change is structural, it is a non-volatile change.

Now it is necessary to discover the relation between w and the charge that flows through the device. To do so, we will use the concept of drift velocity. The drift velocity of a particle is given by

$$v_d = \mu \mathbf{E} \tag{3.2}$$

where μ is the ion mobility and E is the applied electrical field.

The drift velocity is the time derivative of the memristor's width, so we can describe the

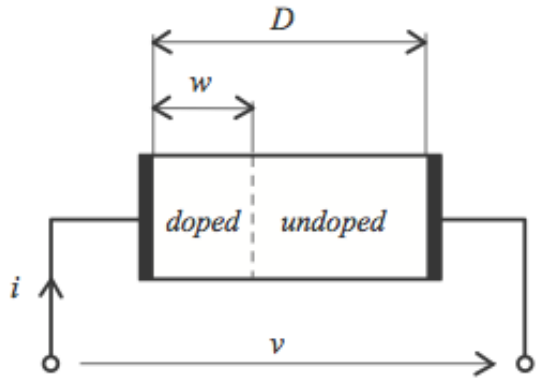

Figure 3.2: Memristor's inner dimensions

system dynamic's by this equation. Rearranging it into a more usable form we get:

$$v_d = \mu \frac{R_{ON}}{D} i(t) \tag{3.3}$$

Right now, we have a couple of equations, which, according to [6], are enough to describe a memristive system. Being the memristor itself a particularization of a memristive system, it can be described using this couple of equations. Let's recall them:

$$v(t) = \left(R_{ON} \frac{w(t)}{D} + R_{OFF} \left(1 - \frac{w(t)}{D} \right) \right) i(t) \tag{3.4}$$

$$\dot{w}(t) = \mu \frac{R_{ON}}{D} i(t) \tag{3.5}$$

Given this, we can create a simulation model in a circuit simulation program, like PSpice, which we will do in Section 3.2. After that we can create circuits with memristors, and observe their behavior in a circuit.

3.2 Model

In order to create a SPICE model, one must choose between two types of models, a sub-circuit and a model. We chose the sub-circuit type. A sub-circuit consists in encapsulating a small circuit, that implements the desired function.

We want to implement a function that returns a resistance value according to its past. So we want to implement Equation 3.1. If we recall that $R_{ON} \ll R_{OFF}$, we can make the following approximation

$$R(w(t)) = R_{OFF} \left(1 - \frac{w(t)}{D} \right) \tag{3.6}$$

If we integrate Equation 3.5, in order to have $w(t)$, and substitute that in Equation 3.6, we get what we should implement:

$$R(w(t)) = R_{OFF} \left(1 - \frac{\mu R_{ON}}{D^2} q(t) \right) \tag{3.7}$$

Where $q(t)$ is the charge that flowed through the device. Given this, we started by implementing the circuit in Figure 3.3.

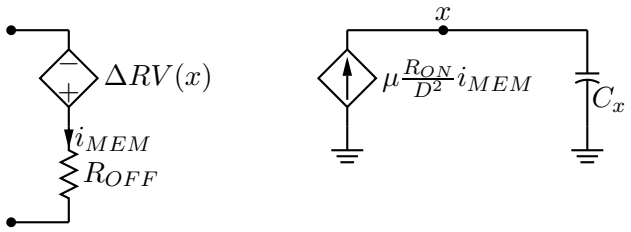

Figure 3.3: Circuit used to simulate a memristor

This circuit uses two controlled sources, one is a voltage controlled voltage source, and the other is a current controlled current source. The current source charges a capacitor, whose voltage is used to change the voltage of the voltage source.

The capacitor is charged using a current source, in order to make its voltage directly depending on the charge that flowed through the device, as shown in Equation 3.8. Solving the equation, we have:

$$v(t) = \frac{1}{C}\int i(t)dt \tag{3.8}$$

$$v(t) = \frac{1}{C}\int \frac{\mu R_{ON}}{D^2} i_{MEM} dt \tag{3.9}$$

$$v(t) = \frac{\mu R_{ON}}{CD^2} q(t) \tag{3.10}$$

If the capacitor has the value of $1F$, we have a term of the Equation 3.7. If we multiply that by R_{OFF}, and then subtract R_{OFF}, we implement Equation 3.7. That is why there is also a voltage source.

This yields in a malfunctioning model, since the device can present a voltage superior to $i_{MEM}R_{OFF}$, and a negative one inferior to $i_{MEM}R_{ON}$. To workaround this glitch, we have to assure that V_{C_x} does not go above, or below, certain values. In order to accomplish this, we have to, somehow, limit V_{C_x}.

We can multiply the value of the current source, with a window function, so it limits the charge or discharge of the capacitor when it approaches the limit values. The window function we used is

$$f(x) = 1 - (2x - 1)^{2p} \tag{3.11}$$

where p is a positive integer, and x is the ratio between w and D.

The form of this window function guarantees zero speed of the coordinate when approaching either limit, therefore ensuring that the boundaries are respected.

In our model implementation, we used some parameters so we can define the memristor's characteristics for each simulation. Those parameters are used to define, R_{ON}, R_{OFF} and the initial resistance R_{init}, D, p and μ.

3.3 Simulations

There are two ways of changing the state of a memristor. Making charge, or a magnetic flux, flow through the device. In this section we will test the created model under these two conditions.

3.3.1 Voltage Controlled Simulation

With this model implemented in PSpice, we created a circuit to test the model. The circuit was the one presented in Figure 3.4.

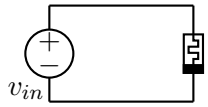

Figure 3.4: Testing circuit for memristor's model

Variable	**Value**
R_{ON}	$1k\Omega$
R_{OFF}	$100k\Omega$
R_{init}	$80k\Omega$
D	$10nm$
μ	$10\frac{fm^2}{Vs}$
p	1
v_{in}	$1.2V$
f_0	$1Hz$

Table 3.1: Memristor model voltage controlled test parametrs

For the first set of tests, we used Table's 3.1 values for the memristor's specifications, and inputs. With these parameters, and the signal in Figure 3.5 applied to the test memristor we obtained the current curve presented in Figure 3.6. We don't obtained a perfect sinusoidal signal, what proves that the resistance of the device is varying along the simulation, and that is depicted in Figure 3.7. This can be seen as a variation of dopant's width in the memristor, and that is depicted in Figure 3.8. This figure is not represented in an absolute measure, but instead is represented in relative terms due to the total memristor width (w/D). Given all this, we are able to see that the voltage current relationship is non-linear, as is depicted in Figure 3.9. As said before, as we increase the frequency, the memristor degenerates in an ordinary resistor, as shown in Figure 3.10, Figure 3.11 and Figure 3.12.

After these tests we tested the window function. To do so, we used the same circuit configuration, but this time with a direct current (DC) source, and we used two values for R_{init}, R_{ON} and R_{OFF}. Starting at these values, the memristor's resistance has to stop on

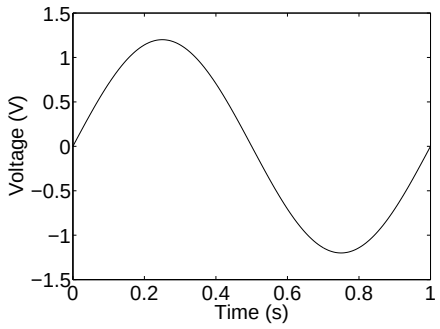

Figure 3.5: Test signal applied to the memristor

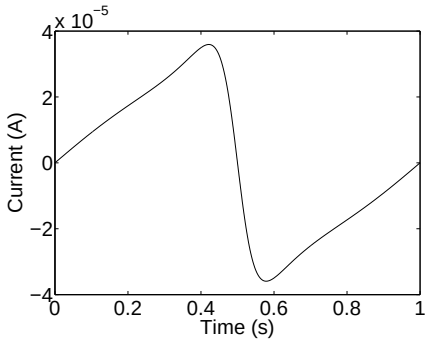

Figure 3.6: Memristor's Current

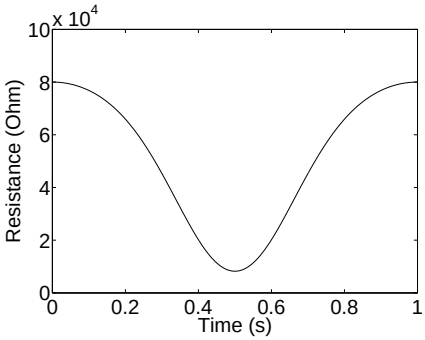

Figure 3.7: Memristor's Resistance

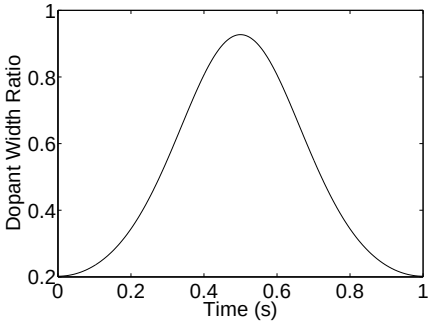

Figure 3.8: Memristor's dopant width

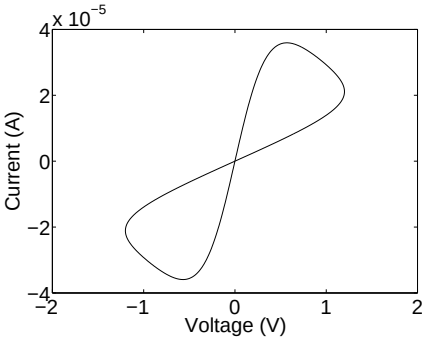

Figure 3.9: Memristor's voltage current characteristic at 1Hz

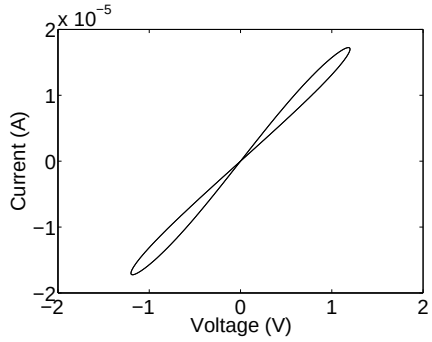

Figure 3.10: Memristor's voltage current characteristic at 2Hz

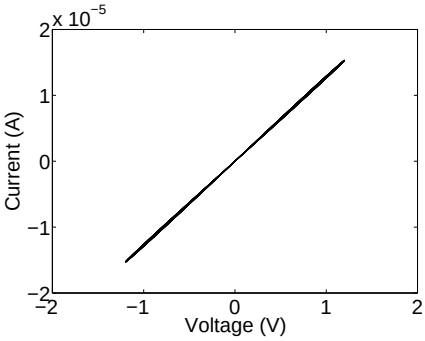

Figure 3.11: Memristor's voltage current characteristic at 10Hz

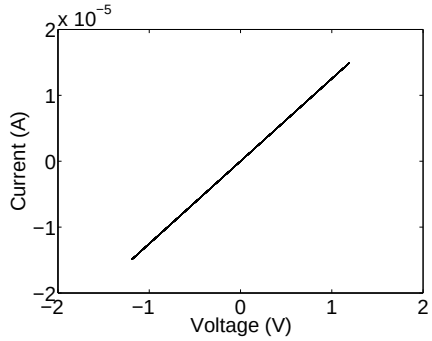

Figure 3.12: Memristor's voltage current characteristic at 100Hz

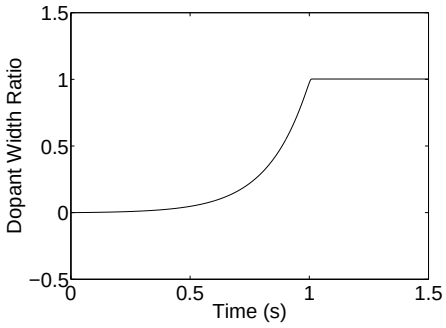

Figure 3.13: Memristor model upper boundary test

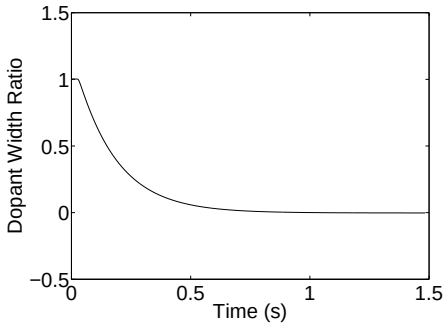

Figure 3.14: Memristor model lower boundary test

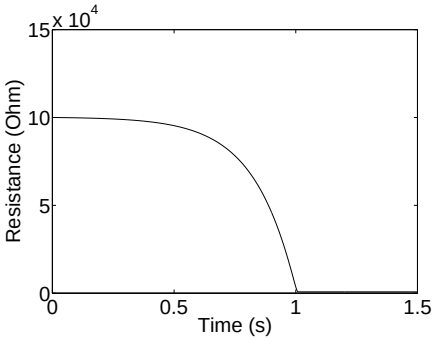

Figure 3.15: Memristor model upper boundary test resistance evolution

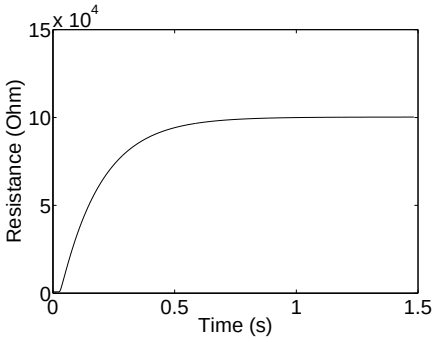

Figure 3.16: Memristor model lower boundary test resistance evolution

the other, when a DC signal is applied.

In Figure 3.13 is represented the evolution of the barrier between the doped and undoped regions of the memristor, when it is switching from the OFF state to the ON state, while Figure 3.14 shows the opposite. Figure 3.15 and Figure 3.16 represent the evolution of the resistance while the device it's switching, to ON, and OFF respectively, while the current that is being consumed in that transition is depicted in Figure 3.17 and Figure 3.18, accordingly.

When the device reaches the boundaries, the voltage current relationship suffers the distortion shown in Figure 3.19.

With these tests,we verified that the created model behaves as a memristor. Yet we can't conclude that the model behaves as expected. To conclude that, we have to test the model with the same parameters that are presented in [4], and compare the results that we obtained with the ones that are presented there.

Those parameters are the ones presented in Table 3.2, followed by the parameters of Table 3.3. Using Table 3.2 parameters we obtained a current curve that is depicted in Figure 3.20, which comes by the barrier change of position, shown in Figure 3.21. This results in the voltage current relationship represented in Figure 3.22. Then, using the values of Table 3.3 we obtained Figure's 3.23 current curve, that is a consequence of the change of the dopant width, which is seen in Figure 3.24. The voltage current relationship is the one shown in Figure 3.25.

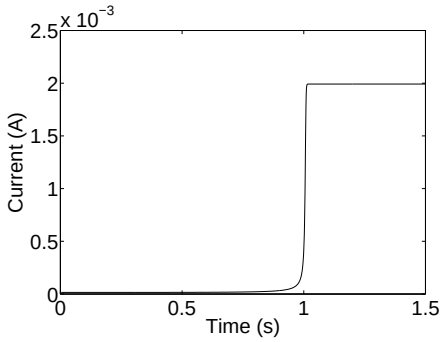

Figure 3.17: Memristor model upper boundary test current

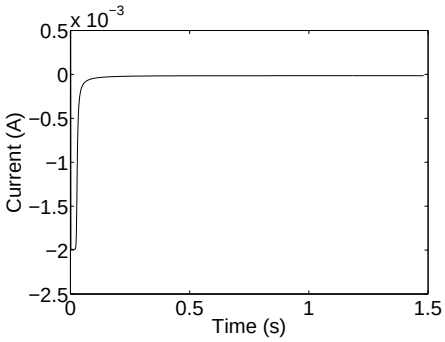

Figure 3.18: Memristor model lower boundary test current

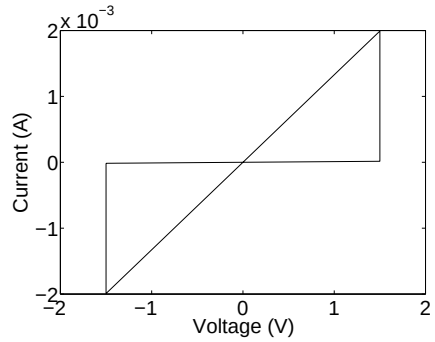

Figure 3.19: Saturated memristor voltage current relationship

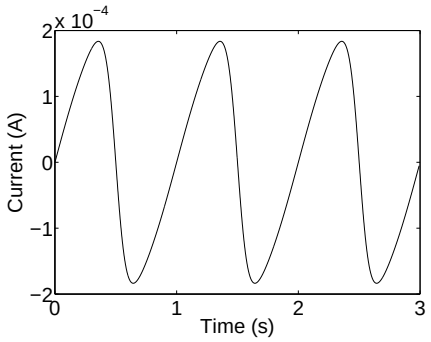

Figure 3.20: Current from testing the model with Table 3.2 parameters

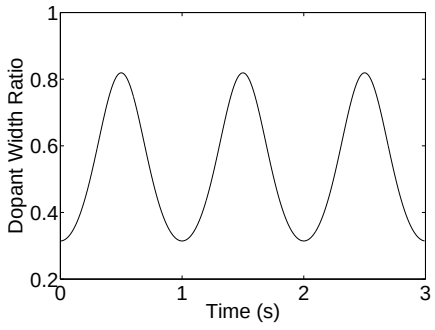

Figure 3.21: Regions ratio from Table 3.2 parameters test

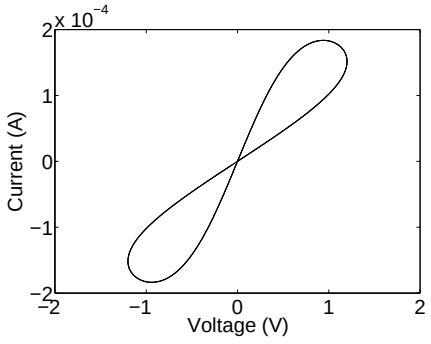

Figure 3.22: Voltage current relation from Table 3.2 parameters

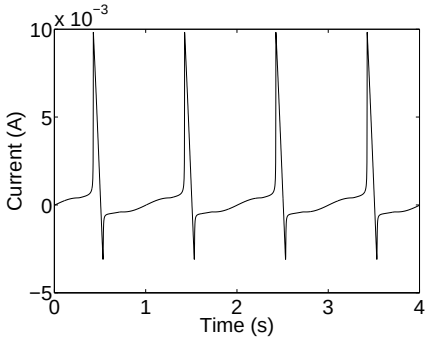

Figure 3.23: Current when testing the device with Table's 3.3 parameters

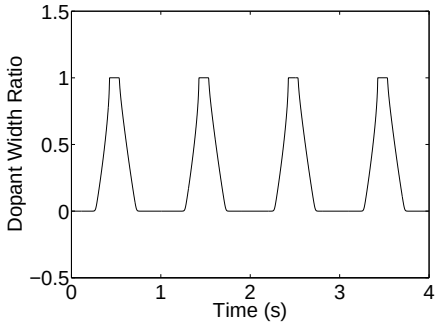

Figure 3.24: Ratio of the two regions when testing using Table's 3.3 parameters

Variable	**Value**
R_{ON}	100Ω
R_{OFF}	$16k\Omega$
R_{init}	$11k\Omega$
D	$10nm$
μ	$10\frac{fm^2}{Vs}$
p	10
v_{in}	$1.2V$
f_0	$1Hz$

Table 3.2: Memristor model verification parameters

Variable	**Value**
R_{ON}	100Ω
R_{OFF}	$5k\Omega$
R_{init}	$1k\Omega$
D	$10nm$
μ	$10\frac{fm^2}{Vs}$
p	10
v_{in}	$2V$
f_0	$1Hz$

Table 3.3: Memristor model test parameters

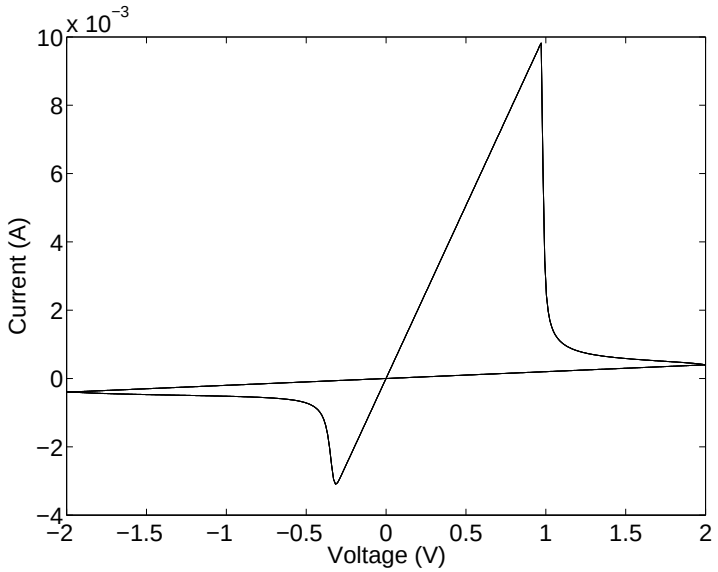

Figure 3.25: Voltage current characteristic when using Table's 3.3 parameters for testing

Variable	Value
R_{ON}	100Ω
R_{OFF}	$10k\Omega$
R_{init}	$10k\Omega$
D	$10nm$
μ	$10\frac{fm^2}{Vs}$
p	10
i_{in}	$100\mu A$
f_0	$0.25Hz$

Table 3.4: Memristor model test parameters for current simulation

Variable	Value
R_{ON}	100Ω
R_{OFF}	$16k\Omega$
R_{init}	$11k\Omega$
D	$10nm$
μ	$10\frac{fm^2}{Vs}$
p	10
i_{in}	$150\mu A$
f_0	$1Hz$

Table 3.5: Memristor's initial parameters used for comparing the two programming modes

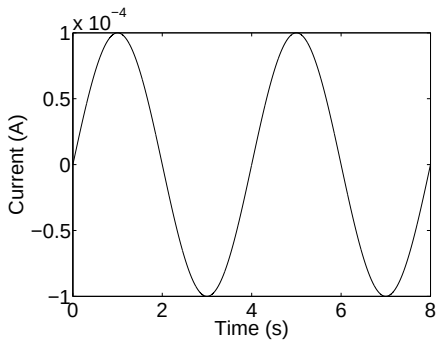

Figure 3.26: Test signal applied to the memristor

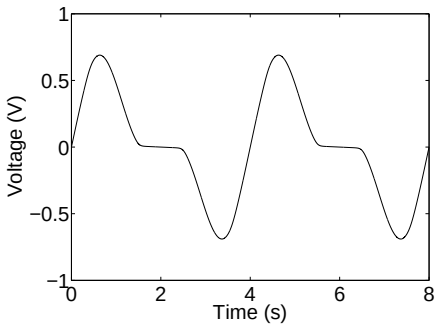

Figure 3.27: Memristor's Voltage

With as much accurate as a reading of [4] graphs can be, we see that what we obtained is quite similar to what we should have. Therefore we can conclude that the model is a completely functional model, with satisfying results.

3.3.2 Current Controlled Simulation

After seeing how this model behaves when tension is applied to it, now we will perform the same type of analysis, but applying current this time.

The circuit we used for this purpose is similar to the one we used for the voltage controlled simulation, but instead of having a voltage source, obviously we have a current source. The memristor parameters we used this time, are the ones of the Table 3.4.

In this simulation, we applied the signal depicted in Figure 3.26, and that resulted in the voltage shown if Figure 3.27. From this we obtained the device's resistance variation along time, which is the curve in Figure 3.29, which is what we expected form the memristor's inner state presented in Figure 3.28.

After this simulation, we performed another one, similar, as much as possible as the one in Table 3.2, which parameters are in Table 3.5, in order to compare the two control methods.

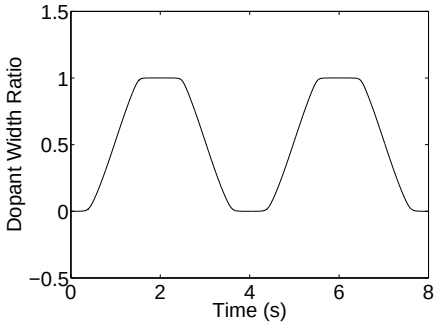

Figure 3.28: Memristor's Inner State

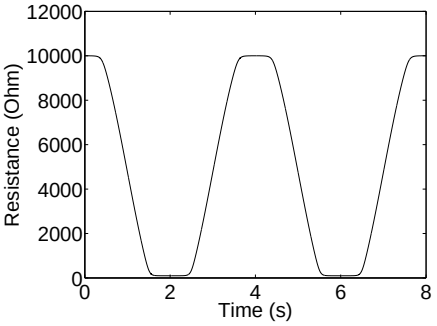

Figure 3.29: Memristor's Resistance

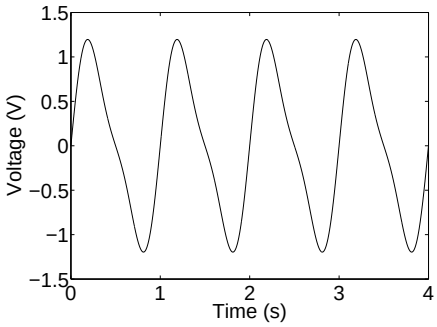

Figure 3.30: Memristor's Voltage

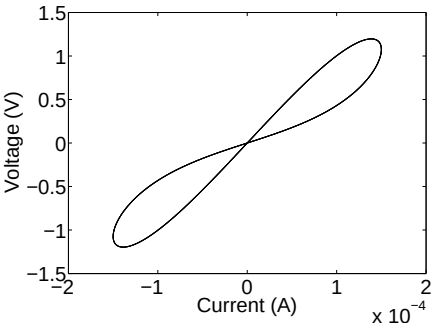

Figure 3.31: Memristor's Current Voltage Relationship

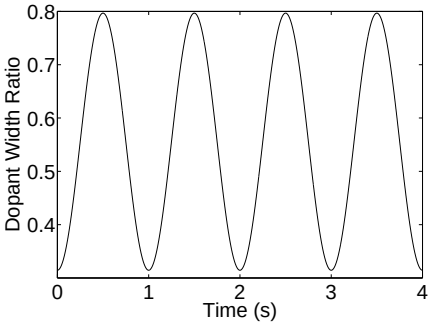

Figure 3.32: Memristor's Inner State

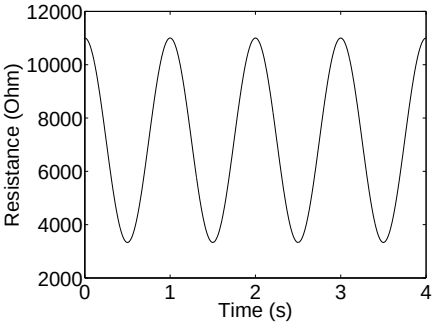

Figure 3.33: Memristor's Resistance

The results from that simulation are the ones presented from Figure 3.30 to Figure 3.33, where we can see that the results are pretty much the dual of those obtained when simulating with the parameters from Table 3.2.

3.3.3 Charge Controlled versus Flux Controlled

So in order to summarize this section, we will discuss the advantages and disadvantages of these two control processes, and compare its ease of use. To this discussion we will have in mind the programming of a value on a memristor, or its switching between states.

The two methods are quite similar, they make the perfect dual, as it was expected since underneath it all, a memristor behaves like a resistor.

The major difference between the two modes is the current that flows through the device. Let's take, for example, the memristor's parameters mentioned in Table 3.4, and assuming that we will work with $100\mu A$ when we want a charge controlled mode, and that we will use $1V$ when we want the flux controlled mode.

If it is a charge controlled switching, the current is fixed, and then the voltage will drop or raise, according to and from which state it is changing. Comparing this current to the current that is yielded in the flux controlled case, this current can be lower at many points of the switching operation. In fact it will only be equal in the lower value case. When we are working in the flux controlled mode, the current can be as much high as 100 times the value we used in the charge controlled mode.

Here, in our simulations, we did not take in account the electrical disruption or thermal constraints of the used material, which could cause problems in real conditions, so we could not say that this current is too high, or the other one is too low. We can say, though, is that with the memristor's dimensions in mind, the power dissipated over the area in the second case, is extremely high, and would probably bring problems.

Despite the fact of the physical constraints the two modes used with the parameters mentioned above, should differ significantly in the switching times, because what changes a memristor's state is the amount of current that flows through it, and here we are talking about a huge difference between the two modes current.

What we have talked above is the main difference between the two programming modes. It could be an advantage or a disadvantage depending on the case.

Other differences are for example the closed loop control of the value that is being stored, if we want to store continuous values, or wait for the switching to complete, in order to start another task. A voltage is easier to sense that a current, so the charge controlled mode wins this battle.

Chapter 4

Memristor Memory Programing

In this chapter we will explain how we program a memristor, and how to build a memory based on these devices.

4.1 Programmer

Now that we studied thoroughly the memristor, we should create a circuit that provide us with a little abstraction on the reading and writing processes of a memristor.

On our agenda we have that we should prove that memristors can be used to store continuous values, and not only the two binary states, as usual. So we should have this in mind when creating our programmer. We should also be aware that we want to store voltage samples, and what we can store are resistance values.

As said previously, to program a memristor, we can do one of two things. Apply a voltage, or a current to the device. If we apply a current, we read a voltage that tells us the memristor's state. If we compare this value to the sample value that we want to store, we know if we should stop the writing process or not. So we achieved our goal of having a method that converts a voltage into a resistance value. In order to ensure consistency, we should make sure that we always have the same current flowing through the device.

Now we are able to create a circuit that programs a continuous value on a memristor. We just need a current source controlled by a comparator.

This is not all. We should also include the capability of reading a memristor's state, and to do so, as we saw earlier, we need to apply an alternating signal with a null mean value, and ensure that the period isn't in the same magnitude as the memristor's switching time. To do so, we should start by including two control *bits*, a $ReadEn$ bit and a $WriteEn$ bit, so we can choose the correct programmer's function. To include these *bits* we need a few logic gates. We need an *and* gate to only care about the comparison when the $WriteEn$ signal is active, and then we need an *or* gate to actuate on the current source, whether we are writing or reading.

But we haven't assured yet that when we read, we do not destroy the stored value. To accomplish that we should use another current source, with opposite direction than the previous one, and when we read, we should alternate, with a fixed period, between them. We need one $Clock$ signal, in order to synchronize them. With this clock a few more logic

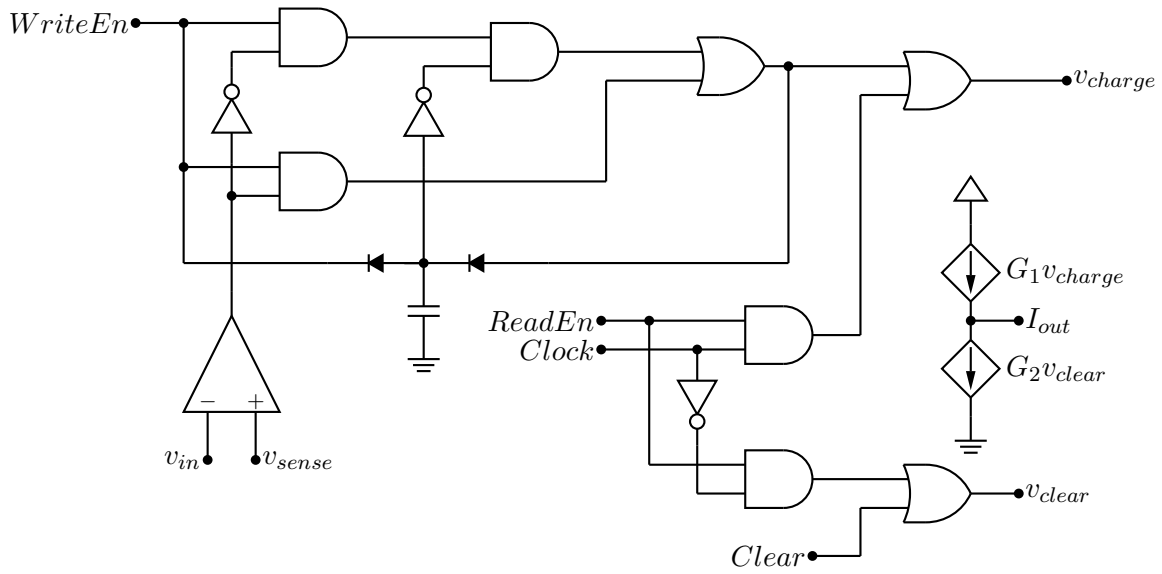

Figure 4.1: Circuit used to program a memristor

$WriteEn$	Comparator	Programmed	Output
0	X	X	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Table 4.1: Truth table for the writing process

gates come to our programmer. Now we should give the previous *or* gate the output of an *and* operation between the *ReadEn* and *Clock* signals, while the new current source will be actuated by the *and* between the *ReadEn* and the *Clock*'s complement.

Now that we are able to read and write on a memristor, what if we want to change its value to another one? We should be able to *clean* the memristor's state in order to start a clean and controlled writing. To accomplish this we need another signal and another *or* gate. We need a *Clear* signal that will actuate on the second current source, but this source already is actuated by a gate output, so we have to put an *or* gate between this *and* and the current source. This gate output should be the *or* comparison between the said *and* gate and the *Clear* signal.

Analyzing the circuit there's only one thing left. We should assure the writing start-up conditions, because if the comparator output is 0, the writing process will not start. On the other hand, if we implement a simple pull-up, we do not stop writing when we should.

Now the things get a little more complex than they were until now. To perform the start-up, and the proper writing finish, we have to implement a function. This function should respect the truth table in Table 4.1, that yields in the following equation. We used letters to create a more compact equation, and W corresponds to $WriteEn$, C to Comparator and P to Programmed.

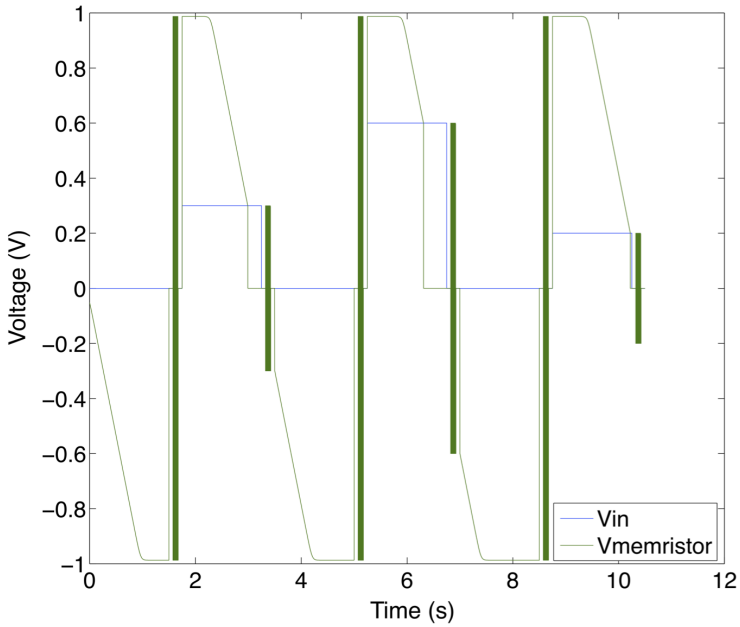

Figure 4.2: Programmer's input signal matched with memristor's voltage

$$\begin{aligned}
\text{Output} &= (W \cdot \overline{C} \cdot \overline{P}) + (W \cdot C \cdot \overline{P}) + (W \cdot C \cdot P) && (4.1)\\
\text{Output} &= W \cdot (\overline{C} \cdot \overline{P} + C \cdot \overline{P} + C \cdot P) && (4.2)\\
\text{Output} &= W \cdot (\overline{C} \cdot \overline{P} + C \cdot (\overline{P} + P)) && (4.3)\\
\text{Output} &= W \cdot (\overline{C} \cdot \overline{P} + C) && (4.4)\\
\text{Output} &= W \cdot \overline{C} \cdot \overline{P} + W \cdot C && (4.5)
\end{aligned}$$

In order to implement this function, we substituted the *and* gate that operated between the $WriteEn$ and the comparator's output for a few more logic gates. For the $W \cdot \overline{C} \cdot \overline{P}$ member, we choose to use two separate gates instead of just one with three inputs, so one gate computes $W \cdot \overline{C}$, and the other one computes the logic *and* between this output and $\overline{P}$. Then this output is the input to an *or* gate along with the output of another gate that solves $W \cdot C$.

To see if the memristor was already written or not, we used a simple level detector, that resets itself when the $WriteEn$ signal goes down. We apply the *not* logic function to the level detector's output, and supply it to the *and* gate that implements the second part of the first member in the equation above.

To finalize this programmer the only thing that needs to be cleared is how the comparison between the memristor's voltage, and the desired voltage is achieved. It can be achieved by a single operation amplifier, with open loop. To achieve the output that we assumed in the

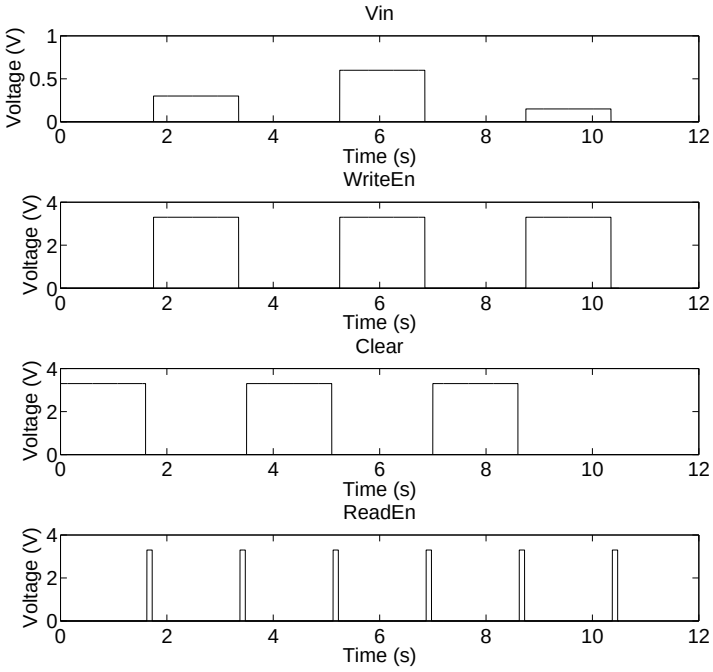

Figure 4.3: Programmer's test signals

truth table above, we need to take in account that when the memristor starts being written, its resistance has the highest value possible, and in our configuration this means that the first moment of v_{sense} is always greater than v_{in}. With this in mind we know that in order to have a null output on the comparator, we should have the v_{sense} on the + *op-amp*'s terminal, while v_{in} should be on the − one.

After having all this in mind, we designed the circuit in Figure's 4.1. With this circuit we can program a memristor, to store any value between R_{ON} and R_{OFF}, as well as read any value that is stored in it. We can also clear the memristor's state.

The current sources values, G_1 and G_2, should be tuned to achieve the desired currents, assuring that when a read cycle is done, the signal that flows through the device has no DC component. To our first test we used 30.3μ for both, since the logic gates should have a maximum output voltage of $3.3V$, and we wanted a current of $100\mu A$.

In order to prove what we explained before, we will run a set of simulations. The test signals are the ones in Figure 4.3, and a clock signal at $100Hz$, with a *dutty-cycle* of 50%, which we don't think necessary to show here.

If we analyze the voltage across the memristor during the test, and match it with the input signal, we can see that the circuit behaves as expected, as shown in Figure 4.2.

Right now we will be explaining the time line present in the test (the signal's form Figure 4.3), in order to understand it. The first thing we do, is clearing the memristor state, to assure that there aren't glitches during the programming stage. Following the erase of the

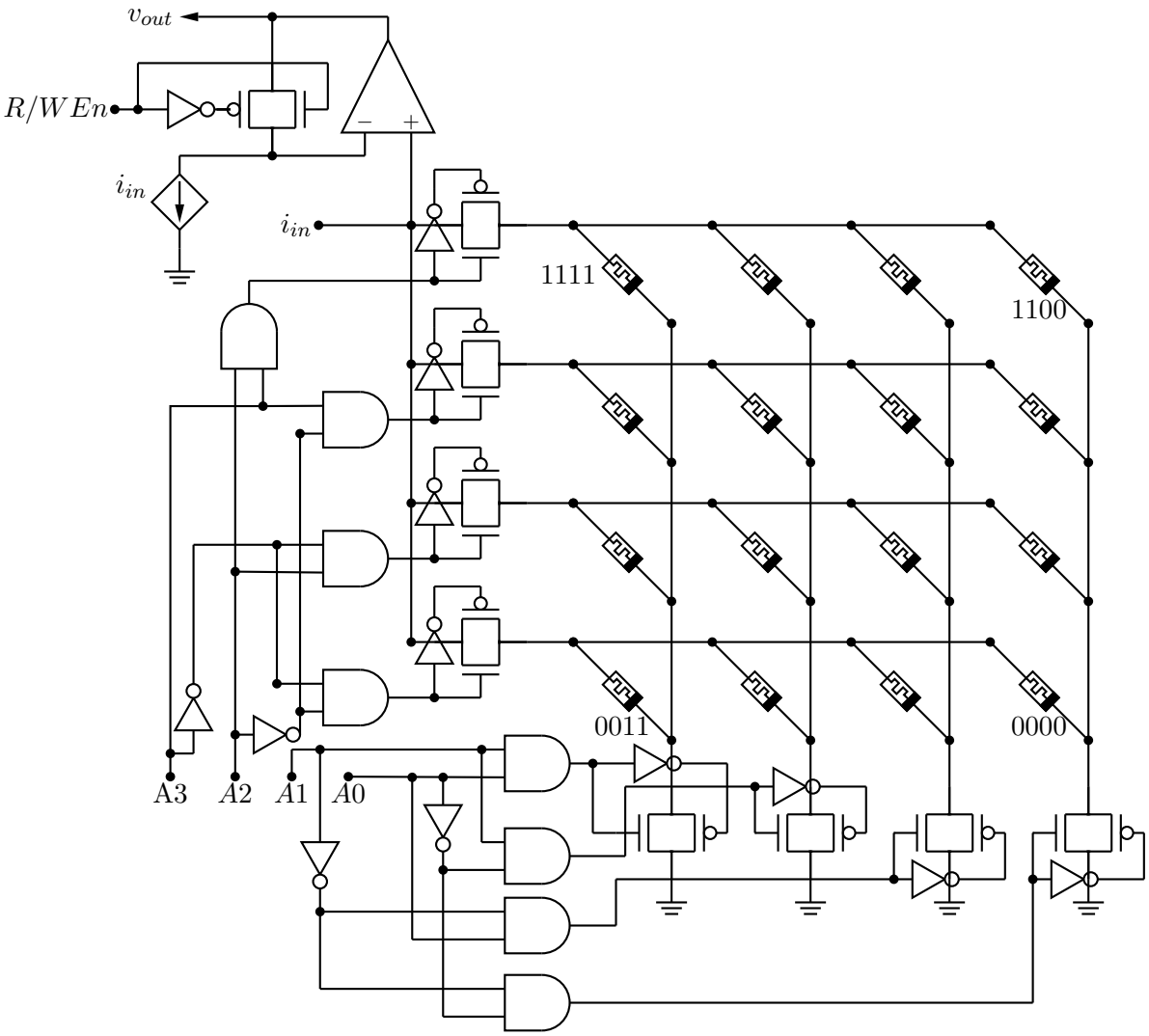

Figure 4.4: Analog Memory Array

memristor, we check that it is properly erased. At this point, we are sure that we have all the conditions to perform a good and successful writing, we proceed to it, checking what has been stored right after the writing process. We repeat this process three times.

It may come to mind a question. Why there is a negative voltage in the output, when we only have positive voltages? The answer is simple. As we have to reset the memristor to ensure a proper writing, we have to make flow a current through it in the opposite direction, hence the negative voltage.

4.2 Memory

Given that we can now program a memristor easily, we can now think of scaling from on memristor to a lot of them.

The memristor's implementation makes it easy to create crossbars arrays[1], that the memristors are on the junctions between the lines and rows of a memory array.

What we are explaining can easily be understood on Figure 4.4, where we have the crossbar architecture, with a line and row address decoder.

In Figure 4.4 we have a little programmer's detail that we don't explained before. In the previous section we said that we compared the memristor's voltage against a reference, but the memristor's voltage was supposedly measured on the same terminal that is used to output the current. That was true if we were programming a single memristor, and that's not the case here.

Here we have a circuit that calculates the amount of voltage that is dropped on the lines and rows address decoder, in order to assure us that we don't make any errors during the reading or writing cycles.

The lithography process of making a memristor crossbar array makes it possible, and easy, to stack several layers of crossbars arrays, creating an incredibly dense non-volatile memory.

If we consider the fact that each memory cell, can store a multi-level, or even continuous, value, this yields in an unprecedented level of denseness.

With such a level of denseness, it is possible to think of using such quantity of memristors to not only store values, but use them for other purposes, like the ones talked in Chapter 2.

[1] Actually the memristors were found when they were trying to implement a crossbar resistive memory, that would save a lot of space when comparing to a transistorized one.

Chapter 5

Linear Combinations

On this chapter we will verify if it is possible to perform linear combinations on the fly, using memristors, and if so, provide some examples of it.

5.1 Operation

A linear combination is a widely used operation, with the following form

$$f = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n \tag{5.1}$$

When we use this operation in electronics, we say that b_i are the inputs, and a_i are weights of that inputs.

If we have the following equation,

$$v_o(t) = 2v_1(t) + 0.5v_2(t) + 5v_3(t) \tag{5.2}$$

this means that $v_o(t)$ will be a linear combination of the inputs, being that the inputs will weigh differently on the output, as it can be seen easily by the equation.

Here we intend to implement this operation quickly enough that it only takes the sampling time to perform the entire operation.

5.2 Simple Implementation Test

If we want to implement a linear combination in *hardware*, we can use a basic operational amplifier configuration, called summer. This configuration is the one depicted in Figure 5.1, and its transfer function, is given by:

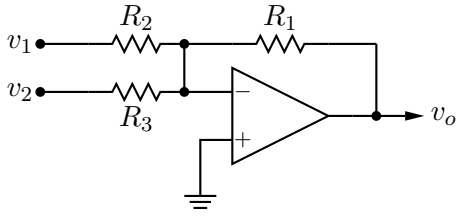

Figure 5.1: Operational Amplifier Summer Configuration

$$v_o = \frac{-R_1}{R_2} v_1 + \frac{-R_1}{R_3} v_2 \tag{5.3}$$

It is important to note that in this case the weights are the resistors ratio. If we want to perform this operation *on-the-fly*, we should be able to define those weights, *on-the-fly* also, which was not possible until we have programable resistors. Now that we have them, we should see if this can actually be accomplished.

To do so we start by testing that memristors can be used to implement a linear combiner, without suffering from any problem, be it the fluctuation of the state, or some non-linearity that is transmitted to the output of the system.

In order to try to maintain the memristor's state unaltered, we should work at frequencies that are above the limit where the memristance mechanism works. On our tests we used $10kHz$, and with our model that proved to be sufficient. It is also important to recall that the minus signs in the equation, will only affect the output's signal phase.

For the first test we used a simple summer configuration, with 4 memristors, that were previously initiated with the desired resistance, to achieve the wished weight for that branch. The circuit used was the one presented in the Figure 5.2.

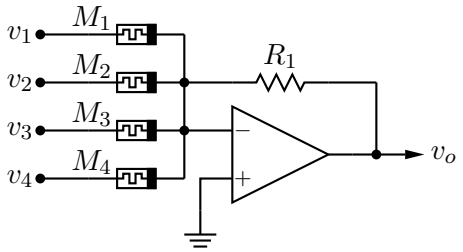

Figure 5.2: Linear Combinator Circuit

Using this circuit, with memristor's values of 500Ω, $2k\Omega$, $1k\Omega$ and 750Ω, from M_1 through M_4 respectively, and R_1 assuming the value of $1k\Omega$, we expect the output to be the following linear combination of the inputs.

$$v_o = -2v_1 - \frac{v_2}{2} - v_3 - \frac{4}{3} v_4 \tag{5.4}$$

If we apply the following voltage sequences to this circuit,

$$v_1 = [1.2 \quad 2.0 \quad 0.7 \quad 1.0 \quad 2.5] \tag{5.5}$$
$$v_2 = [3.0 \quad 1.0 \quad 4.0 \quad 0.6 \quad 2.0] \tag{5.6}$$
$$v_3 = [1.5 \quad 0.3 \quad 1.5 \quad 1.0 \quad 0.2] \tag{5.7}$$
$$v_4 = [1.5 \quad 1.2 \quad 0.5 \quad 3.0 \quad 1.7] \tag{5.8}$$

we expect the following sequence in the output, that is also represented in Figure 5.3, for better comparison of the result.

$$v_0 = [7.4 \quad 6.4 \quad 5.6 \quad 7.3 \quad 8.4] \tag{5.9}$$

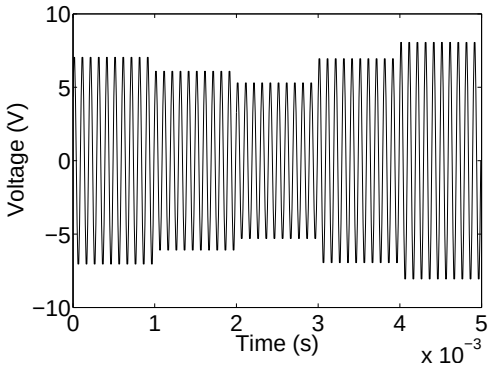

Figure 5.3: Linear Combinator Circuit Expected Output

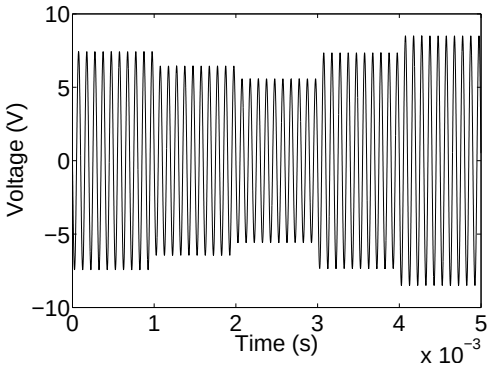

Figure 5.4: Linear Combinator Circuit Simulated Output

In these conditions we obtained the output represented in Figure 5.4, that is exactly what we expected, including the phase inversion of the signal.

With this implementation, we can only make one sign operations, we cannot add one pair of coefficients and subtract the others or *vice-versa*. In order to try and workaround this limitation, we implemented another circuit, the one shown in Figure 5.5.

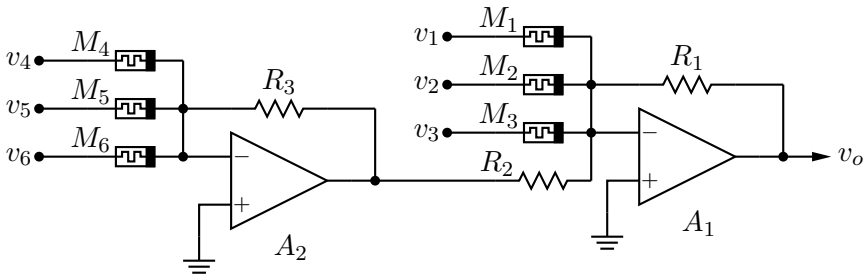

Figure 5.5: Second Approach Linear Combinator Circuit

In this circuit, the inputs from v_1 to v_3 suffer from an inversion of phase, while the inputs v_4 to v_6 don't suffer any change in its phase. The rest of the circuit behavior is exactly like the one explained before.

One thing that must be taken in account is the R_1, R_2 resistor's ratio, since this is important on how the amplifier's A_2 signal is transmitted to the output. It represents the weight of the circuit attached to A_2.

If the memristor's state is initialized in order to obtain the resistance of $2k\Omega$, $3k\Omega$, $1k\Omega$, 800Ω, $3k\Omega$ and $7.2k\Omega$ for memristors M_1 through M_6, and the resistors have the values of $1k\Omega$, $1k\Omega$ and $3k\Omega$ respectively, we expect the output to be

$$v_o = -\frac{v_1}{2} - \frac{v_2}{3} - v_3 + \frac{15}{4}v_4 + v_5 + \frac{5}{12}v_6 \tag{5.10}$$

In order to make a simple test, like we did for the first case, we are going to use the following values for the inputs

$$v_1 = [1.2 \quad 2.0 \quad 0.7 \quad 1.0 \quad 2.5] \tag{5.11}$$
$$v_2 = [3.0 \quad 1.0 \quad 4.0 \quad 0.6 \quad 2.0] \tag{5.12}$$
$$v_3 = [1.5 \quad 0.3 \quad 1.5 \quad 1.0 \quad 0.2] \tag{5.13}$$
$$v_4 = [1.5 \quad 1.2 \quad 0.1 \quad 3.0 \quad 1.7] \tag{5.14}$$
$$v_5 = [0.5 \quad 1.8 \quad 0.5 \quad 0.3 \quad 0.8] \tag{5.15}$$
$$v_6 = [2.3 \quad 0.7 \quad 2.6 \quad 1.4 \quad 1.3] \tag{5.16}$$

so we expect the sequence that is shown in Figure 5.6 and below, in the output

$$v_0 = [4.0 \quad 5.0 \quad -1.2 \quad 10.4 \quad 5.6] \tag{5.17}$$

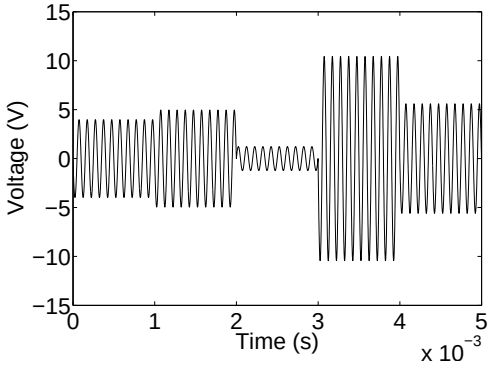

Figure 5.6: Linear Combinator Circuit Expected Output

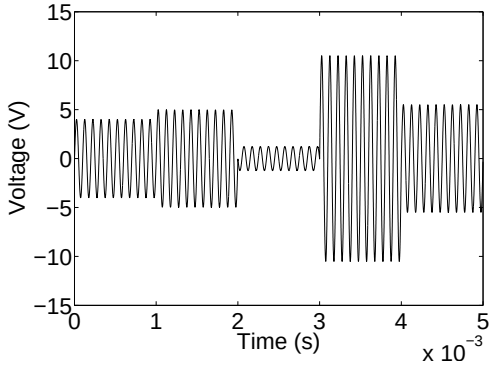

Figure 5.7: Linear Combinator Circuit Simulated Output

We should be aware that since we are working with alternate current, a perfect synchronization of the input signals is mandatory, in order to ensure that there are no miscalculations at any point during the operation.

After we ensure this fact, if we analyze the expected output values, we can see that one value has a different polarity than the others. This is seen in the output results, both theoretical and simulation, when the signal makes a phase change of half wave.

This circuit has yet another limitation. We cannot change a coefficient form positive to negative and *vice-versa*. We are stuck with one of these two parts.

5.3 Changing Coefficients *on-the-fly*

Being proved that linear combinations can be performed in hardware, now we should prove our point. To prove our point we should change the circuit depicted in Figure 5.2, in order to support memristor's programming.

To do so, we have combined that circuit with the one from Figure 4.1, which was properly encapsulated to simplify this schematic, and a few more components to create the circuit of Figure 5.8.

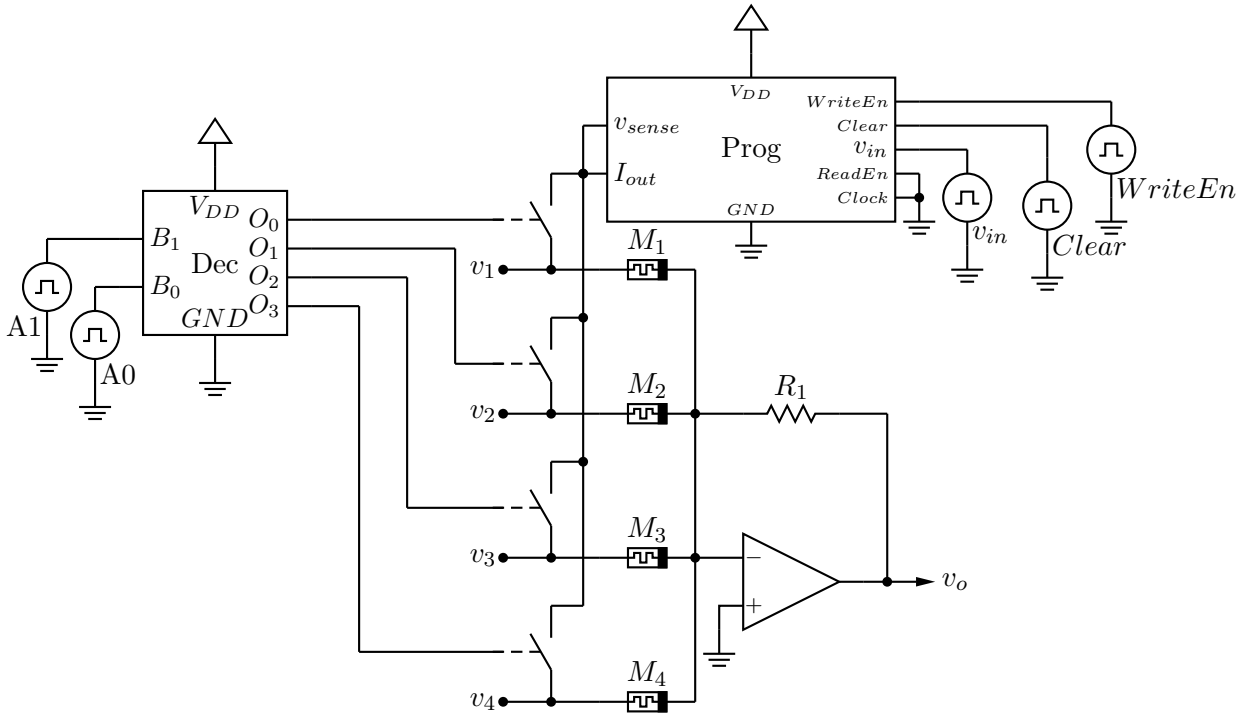

Figure 5.8: Linear Combinator Circuit, for changing the coefficients *on-the-fly*

With this circuit we should be able to configure the memristor's state, and thus change its weight on the operation. To test this, we will use the following weights, respectively, from M_1 through M_4, 2500Ω, $4k\Omega$, 3200Ω and 1500Ω. The feedback resistor, R_1, has the value of $1k\Omega$. To the inputs we will apply $1.5V$, $1V$, 2 and $0.6V$ from v_1 to v_4 accordingly.

With this setup we will have the following equation

$$v_o = -\frac{2}{5}v_1 - \frac{1}{4}v_2 - \frac{5}{16}v_3 - \frac{2}{3}v_4 \tag{5.18}$$

Which yields a $v_o = -1.875V$. We should have in mind that we expect this value with ideal conditions.

When we look at the circuit, and look for points where we expect that will not behave ideally, we find a few components, with the control switches being the most relevant ones.

On our simulation, we defined the switches as having an *off* state resistance of $1M\Omega$, and an *on* resistance of 1Ω. With this in mind we can see that with a memristor being programmed, we have its resistance in parallel with 3 "resistors" with a resistance high enough that the current drained by them should be irrelevant.

This is the normal assumption on similar cases, but here we should have in mind the memristor programming stop conditions. The memristor should stop programming when we reach a certain voltage level, when a pre-determined current flows through it. When we have some resistors in parallel with the memristor, some current of the pre-determined one, will not flow through the device, therefore creating an error on the voltage that is being read.

If we assumed that with a certain current, we should expect a corresponding voltage, in order to know the resistance state of the device. If we change the current value, we change the read voltage, thus changing the resistance final value.

Let's try and see what is the error that we should expect. If we plan to program the memristors M_1 through M_4 with the resistances of 2500Ω,$4k\Omega$, 3200Ω and 1500Ω, with the programmer sourcing $200\mu A$ to the circuit, the stop conditions for the programming are when we reach $0.5V$, $0.8V$, $0.64V$ and $0.3V$, from M_1 through M_4, respectively.

We expect that the voltage at M_1 memristor's terminals should be

$$200\mu A = \frac{v}{2500\Omega} + \frac{3v}{1M\Omega} \tag{5.19}$$

since we know the final voltage and the real current value [1], and we are trying to discover the real resistance, we can solve the following equation

$$61\frac{\mu A}{V} \times 3.3V = \frac{0.5V}{M_1} + \frac{3 \times 0.5V}{1M\Omega} \tag{5.20}$$

$$M_1 = \frac{0.5V}{61\frac{\mu A}{V} \times 3.3V - \frac{3\times 0.5V}{1M\Omega}} \tag{5.21}$$

And we get a resistance value of 2502.5Ω. Applying this calculations to the other memristors we have 4022.1Ω, 3210Ω and 1497Ω from M_2 to M_4 correspondingly.

Rewriting Equation 5.18, we have

$$v_o = -\frac{1k\Omega}{2502.5\Omega}v_1 - \frac{1k\Omega}{4022.1\Omega}v_2 - \frac{1k\Omega}{3210\Omega}v_3 - \frac{1k\Omega}{1497\Omega}v_4 \tag{5.22}$$

If we use the above input voltage values we get the value of $-1.8719V$ for v_o. When we compare this value to results we obtained from the simulation, and feed them to MATLAB® and then run the following command, that is the value that we get (remember that we are working with AC, with none DC component, so the values we are talking are both positive and negative, and they are symmetrical too). The results from the simulation are shown in Figure 5.9, where we can see all programming cycles of the four memristors, and the computed result. Then in Figure 5.10 we have a figure that focus the operation computation.

```
>> max(Vout(:))

ans =
    1.8721

>>
```

We didn't got a perfect match for the expected value, but this doesn't mean that what we tried isn't possible. It only means that we didn't considered some fact. And the value difference is so small that it can be the result of a lack of precision on the simulation results.

[1] We used a trans-conductance gain of $61\frac{\mu A}{V}$ over a voltage of $3.3V$

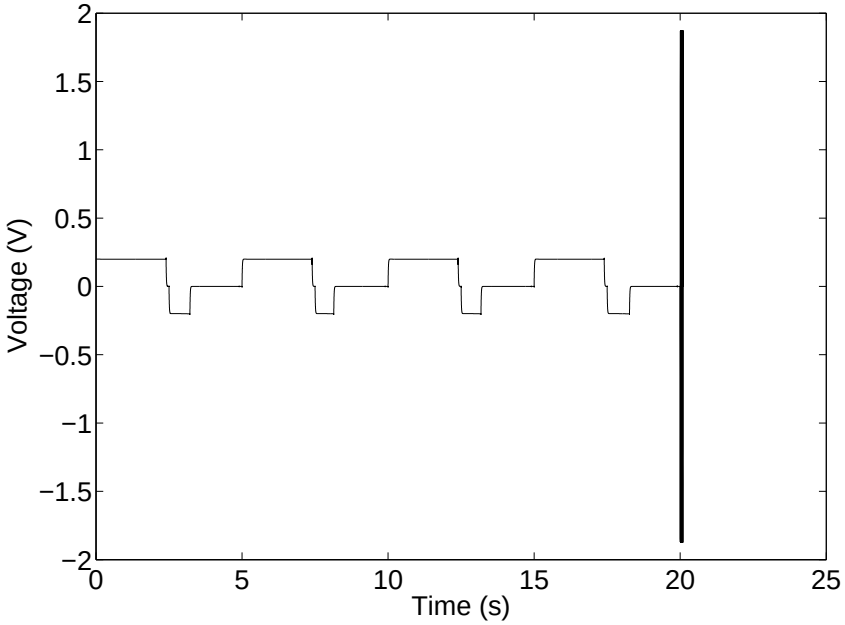

Figure 5.9: Simulated output of the programable linear combiner

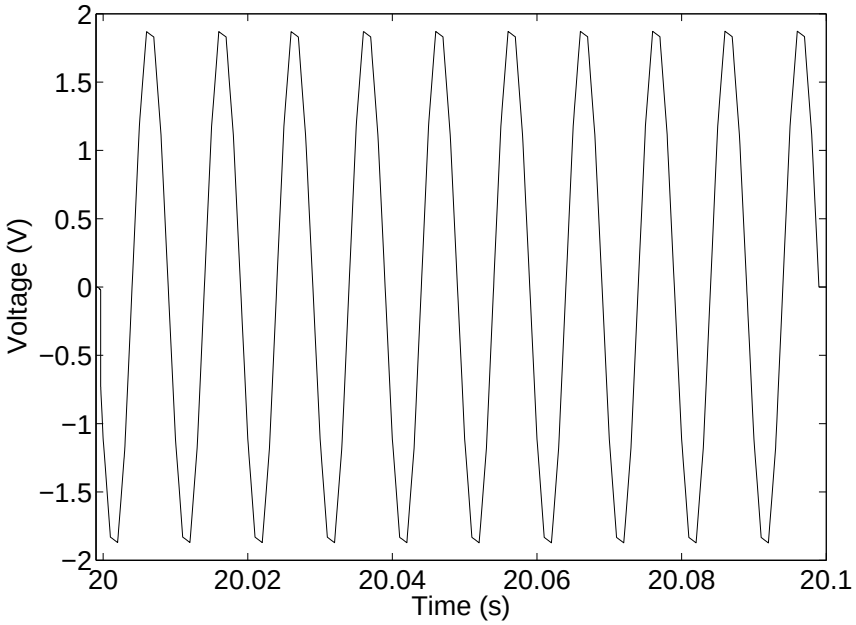

Figure 5.10: Simulated output of the programable linear combiner (focus on the operation)

Chapter 6

Conclusion & Future Work

The main goal of this thesis work, was achieved, as explained in Chapter 5, more precisely in Section 5.3.

With our point proved, we can now move on and try to implement what we discussed in Section 1.1, and Section 2.3, or we can move on and develop what we will talk below.

While we proved our point, we also provided an abstraction of the physical characteristics of a memristor, when it comes to writing or reading it.

This thesis introduces a new working area, an area that will have to be developed from ground up.

To do so, the first thing that must be done is creating a new memristor model. The model presented here is ideal, and does not have the capability of show us the size, power and material's interference in the memristance effect. It would be needed that the created model would work like a memristor simulator itself, giving him the desired parameters like the height, width and depth, the conductor's dimensions and so on, and then the model gives us back the detailed information about its state, temperature across the device, etc.. Some pertinent literature on this matter, was published recently [12].

Another ground work that can, and should, be done is a new programmer. A programmer that does not need the clearing step before writing, and that doesn't need an external clock signal to perform safe readings. Maybe even create a programmer that uses voltage and senses current.

Another interesting work would be to implement the basic boolean logic functions, in order to provide a good background to CMOS replacement and start some new reconfigurable electronic paradigms. A starting point would be [11] and [13].

After this step a fully *memristorized* memory could be implemented. For fully *memristorized* memory we understand a memory that uses memristors to store the pretended data, and uses memristors to create the memory control (row and line decoders).

Now that we are talking in memories, a multi-level memory should also be created, i.e., a memory that would store one of various levels on a single cell. And this could be taken seriously, and to some noise critical applications, such as audio, the number of levels could be infinite, this means, continuous.

Here we arrive at the point where we talk about the future development of the main goal of this thesis. It should be created a circuit that somehow simplifies the process of programing the memristors and then perform a linear combination, that enables this and at the same time provides a good abstraction in order to be scalable. We should also be aware of the limitation mentioned before, in Section 5.2, of a coefficient only can behave as positive or negative at a time, depending to what operational amplifier it is connected, and provide a simple workaround for it.

When the previous goal is achieved, it will be easy to connect several of those circuits in order to perform a meaningful operation, be it a simple signal filter, or an artificial neural network.

Until now we talked about ground work. When this work is done, we have good layer of abstraction to start creating some good products on top of it, and some novel applications also.

After we have all these tools, we can integrate them in a manner to simulate an artificial brain, with areas to store information, areas to process information from the "senses", and areas to perform logic operations, using mainly memristors. That respects one of the basic laws of artificial intelligence, the one that says that the delay between the input and output of a system should not be greater than the propagation time.

Talking in novel applications, we can think in a system that encrypts/decrypts high speed transmissions, like the ones used in optic fiber connections. *Instant-on* computational systems. Improved and quicker beamforming solutions.

Bibliography

[1] CHUA. Memristor - The Missing Circuit Element. IEEE Transactions on Circuit Theory (1971) vol. ct-18 (5) pp. 507 – 519

[2] Williams. How We Found The Missing Memristor. Spectrum, IEEE (2008) vol. 45 (12) pp. 28 – 35

[3] Strukov et al. The missing memristor found. Nature (2008) vol. 453 (7191) pp. 80-83

[4] Biolek et al. SPICE Model of Memristor with Nonlinear Dopant Drift. Radioengineering (2009) vol. 18 (2) pp. 210 – 214

[5] Kavehei et al. The fourth element: Insights into the memristor. International Conference on Communications, Circuits and Systems (2009) pp. 921 – 927

[6] Chua and Sung Mo Kang;. Memristive devices and systems. Proceedings of the IEEE (1976) vol. 64 (2) pp. 209 – 223

[7] http://www.memristor.org/reference/295/types-of-memristors on May, 14^{th} of 2010

[8] Widrow et al. Birth, Life, and Death in Microelectronic Systems. Military Electronics, IRE Transactions on (1961) vol. MIL-5 (3) pp. 191 – 201

[9] Argall. Switching Phenomena in Titanium Oxide Thin Films. Solid-State Electronics (1968) vol. 11, pp. 535 – 541

[10] Borghetti et al. 'Memristive' switches enable 'stateful' logic operations via material implication. Nature (2010)

[11] Raja and Mourad. Digital Logic Implementation in Memristor-Based Crossbars - A Tutorial. Electronic Design, Test and Application, 2010. DELTA '10. Fifth IEEE International Symposium on (2010) pp. 303 - 309

[12] J. P. Strachan, D. B. Strukov, J. Borghetti, J. Joshua Yang, G. Medeiros-Ribeiro, and R. Stanley Williams, "The switching location of a bipolar memristor: chemical, thermal and structural mapping," Nanotechnology, vol. 22, no. 25, p. 254015, May. 2011.

[13] T. Raja and S. Mourad, "Digital Logic Implementation in Memristor-based Crossbars" in 2009 International Conference on Communications, Circuits and Systems (ICCCAS), 2009, pp. 939–943.

[14] E. Nedaaee Oskoee and M. Sahimi, "Electric currents in networks of interconnected memristors," Physical Review E, vol. 83, no. 3, Mar. 2011.

[15] H. Yan et al., "Programmable nanowire circuits for nanoprocessors," Nature, vol. 470, no. 7333, pp. 240–244, Feb. 2011.

[16] K. Eshraghian, K. R. Cho, O. Kavehei, S.-K. Kang, D. Abbott, and S.-M. S. Kang, "Memristor MOS Content Addressable Memory (MCAM): Hybrid Architecture for Future High Performance Search Engines," arXiv, vol. cond-mat.mes-hall. 11-May-2010.

[17] S. Shin, K. Kim, and S. Kang, "Memristor Applications For Programmable Analog ICs," Nanotechnology, IEEE Transactions on, vol. PP, no. 99, pp. 1–1, 2010.

[18] D. Strukov and A. Mishchenko, Monolithically stackable hybrid FPGA. European Design and Automation Association, 2010, pp. 661–666.

[19] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale Memristor Device as Synapse in Neuromorphic Systems," Nano Letters, vol. 10, no. 4, pp. 1297–1301, 2010.

[20] S. Chaudhuri, W. Zhao, J.-O. Klein, C. Chappert, and P. Mazoyer, "Design of Embedded MRAM Macros for Memory-in-Logic Applications," Great Lakes Symposium on VLSI, Proceedings of the 20th symposium on Great lakes symposium on VLS, pp. 155–158, 2010.

[21] H. Kim, M. P. Sah, C. Yang, and L. O. Chua, "Memristor-based Multilevel Memory" in 2010 12th International Workshop on Cellular Nanoscale Networks and their Applications (CNNA 2010), 2010, pp. 1–6.

[22] Y. V. Pershin and M. Di Ventra, "Experimental demonstration of associative memory with memristive neural networks," Neural Networks, vol. 23, no. 7, pp. 881–886, 2010.

[23] T. A. Wey and W. D. Jemison, "An automatic gain control circuit with TiO2 memristor variable gain amplifier," 2010 8th IEEE International NEWCAS Conference (NEWCAS), pp. 49–52, 2010.

[24] M. Stork, J. Hrusak, and D. Mayer, "Feedback systems with memristors," ELECO 2009. International Conference on Electrical and Electronics Engineering, 2009., pp. II–58 – II–62, Nov. 2009.

[25] J. Borghetti et al., "A hybrid nanomemristor/transistor logic circuit capable of self-programming," Proceedings of the National Academy of Sciences of the United States of America, vol. 106, no. 6, pp. 1699–1703, 2009.

[26] D. L. Lewis and H.-H. S. Lee, "Architectural evaluation of 3D stacked RRAM caches," in 2009 IEEE International Conference on 3D System Integration (3DIC), 2009, pp. 1–4.